Professional IPv6 Second Edition

プロフェッショナル

IPv6

第２版

小川 晃通 著

Lambda Note

ここに掲載するスポンサーの皆さまには、
IPv6に関する技術情報を広く公開するという趣旨に賛同いただき、
本書第1版の執筆と制作、公開にあたって多大な協賛をいただきました。

ゴールドスポンサー

株式会社日本レジストリサービス様

BBIX株式会社様

NTTコミュニケーションズ株式会社様

シルバースポンサー

日本ネットワークイネイブラー株式会社様
（本書第1版発行時。現在は株式会社JPIX）

Professional IPv6 Second Edition

by
Akimichi Ogawa

本書は CC-BY-SA-NC ライセンスによって許諾されています。

本書中の会社名や製品名は該当する各社の商標または登録商標です。

序文

「インターネット」は、世界規模の巨大なネットワークです。インターネットを使った通信を行うための仕組みが「インターネットプロトコル」です。多くの人々が「インターネット」として知っている世界規模の巨大ネットワークはインターネットプロトコルバージョン4、すなわちIPv4で作られました。IPv4で使われるIPv4アドレスの数には制約があり、そのうちIPv4アドレスが足りなくなることは予想されていました。その対策として、インターネットプロトコルのバージョン6であるIPv6が開発されました。

IPv6そのものは、決して新しい技術ではありません。IPv6の最初の基本仕様であるRFC 1883[†1]は1995年に発行されています。そのRFCに関連する議論は、それよりも前から開始されています。しかし、IPv6の普及はなかなか進まず、IPv4のみが利用され続ける状況が長く続きました。IPv4アドレスの在庫が枯渇することはかなり昔から予想されており、その対策として作られたIPv6でしたが、IPv4との間で互換性がないこともあって、なかなか普及しなかったのです。

ところが、2011年にIPv4アドレスの中央在庫が現実に枯渇したことにより、IPv6への注目が高まります。IPv6を利用したインターネットも急速に拡大していきました。筆者がIPv6について解説するために資料を集め始めた2011年の段階では、世界中のインターネットユーザが利用しているインターネットプロトコルはIPv4でしたが、2017年の段階ではIPv6普及率が50％を超える地域も登場しています[†2]。2021年には世界全体でのIPv6利用率が3割を超えています。

インターネットにおけるIPv6対応が進んだ大きな要素としては、Apple社が2016年6月からiOSアプリの審査基準としてIPv4に依存するコードを禁止したことも挙げられます。これにより、非常に多くのモバイルアプリケーションがIPv6に対応しました。

普及に伴い、IPv6に関する知識を求めるエンジニアは増え続けています。その一方で、IPv6に関するまとまった情報は、あまり多くありません。本書は、多くのエンジニアがIPv6に関するまとまった情報を得やすくすることを目指して執筆したものです。

---

†1 RFC 1883：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 1995年12月

†2 https://www.google.com/intl/ja/ipv6/statistics.html

混ざりつつも、似て非なるプロトコル

インターネットに接続された機器同士の通信を実現するという視点では、IPv4もIPv6も両方とも同じインターネットプロトコルです。パケットという単位に分かれたデータを届けることや、パケットの宛先や送信元をIPアドレスで示すという部分は、IPv4とIPv6で同じ仕組みです。

IPv6の最大の特徴は、IPアドレスが128ビットで表現されるという点です。IPv4のIPアドレスは32ビットで表現されるので、IPv6はIPv4の4倍のビット数、表現できるIPアドレスは2の96乗倍です。

IPv6は、IPv4と同じ「インターネットプロトコル」なので、IPアドレスを表現するビット長だけがIPv4とIPv6の違いだと誤解されがちです。しかし、その中身を少しでも見ると、IPv6がIPv4とは似て非なるものだとわかります。似ている部分も多いのですが、さまざまな違いがあるプロトコルであり、互いに直接の互換性はありません。

直接の互換性がないので、「インターネットプロトコル」という視点で見ると、IPv4とIPv6とはまったく別々のネットワークになります。とはいえ、すべてが異なるネットワークというわけでもないので、実際の運用ではIPv4とIPv6とで同じ物理回線やサーバを共有するのが一般的です。

さらに、「インターネットは1つである」という考えもあって、DNSによる名前空間など、ところどころで意図的にまぜこぜにされている部分もあります。IPv4だけでも難解でわかりにくいインターネットの仕組みが、IPv4とは似て非なるIPv6が並行運用されることにより、さらにややこしいものになりつつあるのが現状です。

本書では、IPv4とIPv6との似て非なる部分を明確にしつつ、まぜこぜに並行運用されるデュアルスタック環境についても解説します。

背景や経緯の説明

IPv6に関連する仕様の多くは、RFCという形で標準化され、その内容は誰でも読める形で公開されています。しかし、RFCに記述されているのはプロトコルの詳細であることが多く、そのプロトコルがなぜ必要なのかや、他のプロトコルとの位置づけの違いなどに関しては、書かれていないことも少なくありません。そのため、慣れていないと、RFCを読んでも仕様の意味を理解しにくいといえます。本書は、RFCを読むだけでは理解しにくい部分を補うことを目指しています。

さらに、IPv6は、標準化された当初と比べると、その中身が大きく変わっています。本書執筆中も、各種プロトコルの更新に関する議論が続けられています。IPv6そのものだけではなく、IPv6に関連する新しいプロトコルも議論されています。IPv6とその周辺技術は常に更新され続けています。

これだけ変化が激しいIPv6とその関連技術を理解するには、プロトコルそのものの解説だけではなく、各種プロトコルが考案された背景を含めた解説が必要でしょう。本書の執筆にあたっては、この点も重視しています。これまでの背景をしっかり理解しておけば、さらに新しいプロトコルが議論され始めたときなどにも理解しやすくなるはずです。

cc BY NC SA

無償配布

本書は、紙の書籍を販売する一方で、紙の書籍と同じコンテンツを電子版として無償配布します。これは、本書企画段階で筆者の知る限りではIPv6に関するまとまった確かな情報がなく、それならばIPv6に関する情報を広くたくさんの人々が知る機会を本書で作りたいと考えたためです。

IPv6は、IPv4と異なる部分も多いので、ちょっとした勘違いが運用上の問題を発生させることもあります。IPv6に関する情報へのアクセスを容易にすることによって、そういった問題の発生を少しでも防げればという想いもあります。

無償配布版の「プロフェッショナルIPv6」は、クリエイティブコモンズライセンスのBY-NC-SA（表示 - 非営利 - 継承）のライセンスのもとに公開します。たとえば、この本に含まれる文章や図を、学校の授業や企業での研修などを目的として複製する場合には、特に申請していただかなくても大丈夫です。それらの用途のために、発表資料や配布資料の中で文章や図を使ったり、PDF本体を複製して組織内で配布していただくことに関しても、特別な許可は必要ありません。

PDFの取得先は、本書の出版元であるラムダノート株式会社のWebサイトにてご確認ください。

- https://www.lambdanote.com/products/ipv6-2

IPv6に興味を抱いている一人でも多くの方の手に本書が届けば幸いです。

本書の構成

本書は5部構成になっています。

第I部は「インターネットとIPv6の概要」というタイトルで、IPv6の視点からインターネット自体の仕組みを復習し、そのうえで、詳細の説明に入る前に把握しておくべきIPv6の概要として、次のような事項を解説しています。

- 従来のIPv4アドレスとは大きく異なるIPv6アドレスの表記や種類
- インターネットプロトコルとしてのIPv6の機能
- IPv4との相違点のまとめ
- IPv6に対応するとはどういうことか
- IPv4とIPv6が混在するインターネットの姿

第II部では、IPv6プロトコルに関して詳しく解説するとともに、その周辺技術をまとめています。本書執筆で最も時間をかけた部分であり、本書の心臓部にあたります。IPv6アドレスの自動設定やマルチプレフィックスに関連する章は特に執筆に注力した部分です。

第III部は、IPv6とIPv4のデュアルスタックに関する解説です。IPv4とIPv6の名前解決を両方とも同時に扱い、ユーザがIPv4とIPv6の違いを認識せずに使えるようにするため、DNSがどのような役割を果たしているかを見ていきます。IPv6インターネットとIPv4インターネットという、直接の互換性がない異なる2つのインターネットが1つになっている構造を解説し

CC BY NC SA

ています。

第IV部は、IPv4アドレス在庫枯渇問題に関連する内容です。IPv6を理解するうえでIPv4の知識が必須ということはありません。しかし、IPv4アドレス在庫枯渇問題は、インターネットそのものを理解するうえで避けては通れない話題であると筆者は考えています。この第IV部では、IPv4アドレス在庫枯渇問題そのものだけではなく、IPv4 NATについても解説しています。IPv4 NATによるIPv4アドレス共有技術は、後に解説するIPv4とIPv6の共存技術でも多く利用されます。

第V部は、IPv4とIPv6の共存技術です。IPv6とIPv4には直接の互換性がないので、さまざまなIPv4/IPv6共存技術が周辺技術として提案されています。過去に提案された手法が改良されて新しい手法として提案されることもあります。この章では、過去の歴史的経緯も含めてさまざまなIPv4/IPv6共存技術を紹介しています。

本書の最後は日本国内に限定した内容を付録として解説しています。付録Aでは、NTT NGNにおけるIPv6について解説しています。NTT NGNにおけるIPv6は、特に日本国内でIPv6について語るときには大きな要素です。この付録の内容を理解するには、第II部のデフォルトIPv6アドレス選択およびマルチプレフィックス、マルチホーム問題に関する章の内容を先に読むことをお勧めします。付録Bでは、日本国内で流通しているCPE（Customer Premises Equipment）で利用されることが多いIPv6パススルーとND Proxyを紹介しています。

本書は上記のような構成になっていますが、その執筆は、正直なところ、途中で断念したくなるぐらい大変なものでした。IPv6の解説書があまり世に出ていないのは、本気で書こうと思うと限りなく書くべきことが見つかるからかもしれません。筆者がIPv6に関する記事を書き始めたのは2011年のことですが、それなりに時間をかけて書き続けてきたにもかかわらず、こうして書籍として公開できる形にできたのはようやく2018年のことです。その間、IPv6を取り巻く世界の状況はかなり変化し、仕様もどんどん変化していきました。書けば書くほど、「あ、これも書かないと……」という発見があり、扱うテーマも雪だるま式に増えていきました。2017年にクラウドファンディングを立ち上げてからは、ブログもあまり書かずに本書の執筆に全力でコミットし、何とか書籍としてまとまった状態になりました。それでも、まだまだ書ききれていない部分があります。たとえば、ルーティングプロトコル、マルチキャストルーティングプロトコル、モバイルIP、IPsecについては、仕様の詳細を解説できていません。それ以外のトピックについても、解説が深い部分と、ある程度さらっと紹介するだけに留まっている部分と、ところどころ濃淡が残っています。

本書を出版しているラムダノート社は、アップデートしていく書籍を目指している変わった出版社です。2018年に、ひとまず一通り形にまとめて初版の出版となりましたが、その後も更新することが当時の目標のひとつでした。そして、2021年12月に第2版として各所を更新したものを出版できました。今後も可能であれば追加や更新を続けていきたいと考えています。

2021年12月

小川晃通

CC BY NC SA

目次

序文 iii

第 I 部　インターネットと IPv6 の概要 1

第 1 章　インターネット概要 3

1.1 IP はパケット交換技術 4
1.2 層に分かれるネットワーク 21
1.3 トランスポート層の役割 25
1.4 オープンなプロトコルと RFC 28

第 2 章　IPv6 概論 37

2.1 IPv6 アドレス 37
2.2 IPv6 アドレスのテキスト表記 38
2.3 IPv6 アドレス体系 43
2.4 IPv6 におけるユニキャストアドレスの構成要素 45
2.5 IPv6 基本プロトコル 45
2.6 ICMPv6 の役割 46
2.7 近隣探索プロトコル 46
2.8 IPv6 アドレスの自動設定 47
2.9 1 つのネットワークインターフェースに複数の IPv6 アドレスが設定される 48
2.10 IPv6 と IPv4 の違い 51
2.11 IPv6 対応とは 54
2.12 IPv6 インターネットと IPv4 インターネットを同時に使う 58
2.13 IPv4 と IPv6 の共存技術 61

第 II 部　IPv6 プロトコルとその周辺技術 65

第 3 章　IPv6 アドレス体系 67

3.1 IPv6 アドレスの種類 67
3.2 IPv6 アドレス空間の使い方は IANA が管理している 69
3.3 IPv6 アドレスのスコープ 70

3.4 IPv6 ノードに要求される IPv6 アドレス 71
3.5 リンクローカルユニキャストアドレス 74
3.6 スコープのゾーン 76
3.7 グローバルユニキャストアドレス 80
3.8 ULA（Unique Local IPv6 Unicast Addresses） 81
3.9 IPv4-Mapped IPv6 アドレス 85
3.10 IPv4-Compatible IPv6 アドレス 88
3.11 例示用 IPv6 アドレス 88
3.12 ユーザへの IPv6 アドレス割り当て 89

第 4 章　IPv6 パケットの構成　93

4.1 IPv6 ヘッダの各フィールド 93
4.2 上位層でのチェックサム計算に使う仮想ヘッダ 103
4.3 IPv6 拡張ヘッダ 105

第 5 章　ICMPv6　115

5.1 ICMPv6 フォーマット 116
5.2 ICMPv6 エラーメッセージ 118
5.3 ICMPv6 情報メッセージ 124

第 6 章　近隣探索プロトコル　127

6.1 近隣探索プロトコルの機能と利用するメッセージ 128
6.2 ルータとプレフィックス情報の発見 129
6.3 リンク層アドレスの解決と近隣不到達性の検知 136
6.4 Redirect メッセージ 143
6.5 近隣探索メッセージのオプション 145
6.6 IPv6 における on-link と off-link 154

第 7 章　IPv6 アドレスの自動設定　159

7.1 SLAAC の流れ 160
7.2 小規模ネットワークにおける SLAAC の利用例 161
7.3 リンクローカルアドレスの生成 162
7.4 SLAAC におけるインターフェース識別子の生成方法 164
7.5 Temporary IPv6 アドレス 166
7.6 DAD（Duplicate Address Detection） 169
7.7 グローバルアドレスの生成 172
7.8 IPv6 アドレスの存続期間とステート 173
7.9 Router Advertisement メッセージによる DNS 情報の配送 177

(cc) BY NC SA

第 8 章 DHCPv6 179

8.1 IPv4 の DHCP と DHCPv6 の違い 180
8.2 IPv4 の DHCP 181
8.3 DHCPv6 の概要 183
8.4 DUID 191
8.5 ステートレス DHCPv6 193
8.6 ステートフル DHCPv6 196
8.7 DHCPv6-PD 201

第 9 章 IP フラグメンテーション 205

9.1 IPv4 におけるフラグメンテーション 205
9.2 IPv6 フラグメントヘッダ 208

第 10 章 Path MTU discovery 215

10.1 Path MTU discovery に関係する IPv4 の機能 216
10.2 Path MTU Discovery に関係する IPv6 の機能 217
10.3 上位層プロトコルと Path MTU Discovery 217
10.4 Path MTU Discovery の仕組みを悪用した DoS 攻撃 218

第 11 章 IPv6 マルチキャスト 221

11.1 IPv6 のマルチキャストアドレス 222
11.2 マルチキャストのスコープ 224
11.3 Solicited-Node マルチキャストアドレス 225
11.4 マルチキャストにおけるゾーン 226
11.5 MLD（Multicast Listener Discovery） 227
11.6 ルータを越えるマルチキャスト 234
11.7 MRD（Multicast Router Discovery） 234
11.8 リンク層でのマルチキャストアドレス 236

第 12 章 IPv6 エニーキャスト 239

12.1 IPv4 におけるエニーキャスト 239
12.2 IPv6 のエニーキャスト 240
12.3 IPv6 サブネットルータエニーキャストアドレス 241
12.4 ユニキャストとしての利用を避けるべき IPv6 アドレス 241
12.5 エニーキャストの注意点 242
12.6 IPv6 エニーキャストと BCP 38 243

第13章 IPv6におけるマルチアドレスとマルチプレフィックス 245
13.1 IPv6アドレスの選択 246
13.2 マルチプレフィックスによるマルチホームの問題 251
13.3 IPv6サイトリナンバリング 257

第14章 IPv6とセキュリティ 263
14.1 IPv6はIPv4よりもセキュアというわけではない 263
14.2 近隣探索プロトコルとセキュリティ 265
14.3 SEND（SEcure Neighbor Discovery） 267
14.4 不正なRouter Advertisementメッセージ 272
14.5 DHCPv6-Shield 275
14.6 IPv6アドレスとプライバシー 275
14.7 IPv6サブネットに対するスキャン 278
14.8 IPsec 278
14.9 ICMPv6を無条件にすべてフィルタリングすべきではない 279
14.10 トンネル技術が抱える問題 280
14.11 IPv4-Mapped IPv6アドレスの問題 281
14.12 ブラックホール用IPv6アドレス 281

第15章 プログラマにとってのIPv6対応 283
15.1 Socket APIとIPv6 283
15.2 単なるIPv6対応では不十分な場合 287
15.3 Happy Eyeballs 288
15.4 IPv6ソケットとIPv4-Mapped IPv6アドレス 292
15.5 ポリシーテーブルの実装 294

第III部 DNSとIPv6 295

第16章 DNSの基礎とIPv6対応 297
16.1 DNSの仕組み 297
16.2 DNSサーバへの再帰問い合わせと反復問い合わせ 298
16.3 DNSメッセージフォーマット 300
16.4 IPv4とIPv6アドレスの問い合わせ例 306
16.5 DNSの逆引き 309
16.6 DNSメッセージの512オクテット問題 313
16.7 IPv6環境におけるDNSの運用上の注意点 319
16.8 廃止された仕様 321

CC BY NC SA

第17章　DNSによるデュアルスタック環境の実現と運用 323

17.1 デュアルスタック環境の実現 323

17.2 IPv6からIPv4へのフォールバック 326

17.3 キャッシュDNSサーバとCDNに関する問題 327

17.4 デュアルスタック環境におけるSRVの利用 328

17.5 IPv6 DNSホワイトリスティングとブラックリスティング 329

17.6 アプリケーションのIPv6対応とデュアルスタック環境 331

第IV部　IPv4はどうなるのか 333

第18章　IPv4アドレス在庫枯渇とその解決策 335

18.1 IPv4アドレスの在庫枯渇はどのような問題なのか 335

18.2 IPv4アドレスの在庫枯渇の影響 336

18.3 IPアドレス管理の階層構造から見たIPv4アドレスの在庫枯渇 337

18.4 IPv4アドレス在庫枯渇の対策 340

18.5 IPv4アドレス移転、IPv4アドレス売買、IPv4アドレス市場 343

第19章　IPv4アドレス共有技術 347

19.1 NATとプライベートIPv4アドレス 347

19.2 一般的なIPv4 NATの仕組み 350

19.3 NAT機器に要求される挙動 356

19.4 IPv4アドレス共有技術の課題 362

第20章　STUNとTURN 371

20.1 STUN 371

20.2 旧STUNにおけるNATの分類 373

20.3 TURN 376

第V部　IPv4/IPv6共存技術 379

第21章　IPv4/IPv6共存技術の分類 381

21.1 IPv4/IPv6共存技術のバリエーション 381

21.2 ステートフルとステートレス 382

21.3 IPv4/IPv6共存技術利用のパターン 382

21.4 今後も多くの試みが誕生する 383

第 22 章 トンネル技術 385

22.1 6to4 ... 386
22.2 Teredo ... 392
22.3 ISATAP ... 399
22.4 6rd ... 402
22.5 4rd ... 404
22.6 6PE ... 405
22.7 トンネル技術とセキュリティ ... 405

第 23 章 IPv4/IPv6 変換技術 407

23.1 IPv4/IPv6 変換の枠組み ... 407
23.2 SIIT ... 410
23.3 NAT64 と DNS64 ... 411
23.4 IPv4/IPv6 変換機用 IPv6 アドレス ... 417
23.5 ALG ... 418

第 24 章 IPv4/IPv6 共存技術の運用形態 421

24.1 DS-Lite ... 421
24.2 Lightweight 4over6（lw4o6） ... 422
24.3 A+P ... 423
24.4 MAP-E、MAP-T、4rd ... 425
24.5 CGN を利用して徐々に IPv6 対応していく方法 ... 430
24.6 464XLAT ... 430

第 25 章 プロキシ方式 433

25.1 HTTP プロキシ ... 433
25.2 TRT 方式 ... 434
25.3 SIP 用 IPv6 プロキシ ... 434

付録 A NTT NGN での IPv6 437

A.1 NTT 閉域網 IPv6 フォールバック問題 ... 437
A.2 NTT NGN IPv6 マルチプレフィックス問題 ... 440
A.3 IPv6 PPPoE と IPv6 IPoE ... 443

付録 B IPv6 パススルーと ND Proxy 455

B.1 ND パススルー ... 455
B.2 ND Proxy ... 455
B.3 IPv6 パススルーや ND Proxy とセキュリティ ... 456

第 1 版へのあとがき **459**

第 2 版へのあとがき **460**

クラウドファンディングにて本書第 1 版をご支援いただいた皆さま........ 461

索引 **465**

CC BY NC SA

更新履歴

各刷における主な修正点および追加点は下記のとおり。その他の変更点に関してはラムダノートのWebサイト（https://www.lambdanote.com）の本書のページを参照してください。

■ 第1版第2刷（2019年2月）

- TURNの解説を追加（第1版 第26章）

■ 第2版第1刷（2021年12月）

- 第Ⅰ部にIPv6のダイジェストを新設
- IPv4アドレス在庫枯渇への対策という視点のうち、「IPv4上での解決」にまつわるトピックを第Ⅳ部に集約
- 図を全体的にブラッシュアップ
- IPv6パススルーとRA Proxyを解説した付録Bを新設
- 各種RFCの改訂に追随
    - IPv6における逆引きDNSに関するRFC 8501
    - RFC 8585へのreferを追加
    - RFC 8504でRFC 8106がMUSTに変わったことを紹介
    - STUN RFC更新に追随。RFC 5389→RFC 8489
    - RFC 8415（DHCPv6）に対応
    - SIIT RFC更新に追随。RFC 6145→RFC 7915
- 説明を追加および補強
    - リンクローカルアドレスの説明を追加
    - Temporary IPv6アドレスと、preferred/valid-lifetimeの説明を補強
    - インターフェースに複数のIPv6アドレスが設定される状況の例示
    - ULAの説明を補強
    - RFC 4007に関連する説明を補強
    - on-linkに関連する説明を補強
    - IPv4ヘッダとIPv6ヘッダの比較説明
    - IPv4 NATの仕組み解説
    - 廃止されたIPv4-Compatible IPv6アドレスの説明
    - NAT66に関して
    - RFC 8106にあるRNDSSでのリンクローカルアドレスに関して
    - RFC 4864にあるNATのSFIについてコラムを追加
    - IPv6アドレスのテキスト表記の名前に関するコラムを追加
    - IPv5に関するコラムを追加
    - 付録AのVNE事業者情報を更新

■ 第2版第2刷（2023年3月）

- IANAで予約されているIPv6アドレスの情報を追記（3.2節）
- 自動生成されたIPv6アドレスを状態について整理し、Optimistic DADに言及（7.8節）
- プログラミングに関する説明とコードの修正（第15章）
- DNSのTCPフォールバックとANYの説明を補足（第16章）
- MAP-Eのインタフェース識別子をRFC 7597準拠に（第24章）

第I部

インターネットとIPv6の概要

ここでは、まずIPv6の視点からインターネット自体の仕組みを復習します。そのうえで、詳細の説明に入る前に把握しておくべきIPv6の概要として、次のような事項を学びます。

- 従来のIPv4アドレスとは大きく異なるIPv6アドレスの表記や種類
- インターネットプロトコルとしてのIPv6の機能
- IPv4との相違点のまとめ
- IPv6に対応するとはどういうことか
- IPv4とIPv6が混在するインターネットの姿

[書籍版では空白のページです]

BY NC SA

第1章

インターネット概要

インターネットという名前は、「〜間の」という意味を持つ「inter」という単語と、ネットワークである「net」を組み合わせたものです。複数のネットワーク同士をつないで相互通信を行うには、どのようなデータをどのような方法で送受信するのかを決めておく必要があります。そういった取り決めのことを**プロトコル**（protocol）と呼びます。

インターネットを実現するためのプロトコルがインターネットプロトコル（IP、Internet Protocol）です。本書で解説する**IPv6**は、インターネットプロトコルのバージョン6というわけです。

IPv6が開発された背景には、IPv4（インターネットプロトコルのバージョン4）アドレス在庫枯渇問題があります（4ページのコラムを参照）。ここで**アドレス**というのは、文字どおり住所や宛先の意味です。インターネットプロトコルで使う住所や宛先は**IPアドレス**といいます。

IPv4では、IPアドレスとして、長さが32ビットの値を使うことになっています。32ビットで表現可能な値の数は、2の32乗通りです。2の32乗は42億9496万7296なので、IPv4アドレスの理論上の総数は約43億個となります。「IPv4アドレス空間の大きさは32ビットである」とか「IPv4アドレス空間からは最大で約43億個のアドレスが割り当て可能である」という表現は、このことをいっています。

IPv6では、IPアドレスを128ビットで表現します。これは、32ビットのIPv4アドレスに対し、長さで見れば4倍です。しかし、それによって利用可能になるアドレス空間の大きさは、IPv4のときの4倍どころではありません。2の128乗通り、すなわち、IPv4アドレスに比べて「2の96乗倍」にもなります。この非常に大きなIPアドレス空間を利用できるようにすることこそが、IPv6の最大の特徴であり、IPv6が開発された理由です。

IPv4アドレスの在庫枯渇問題

インターネットが世界的に普及すると、利用者数の増加に伴ってインターネットに接続する端末の数も増えました。インターネットに接続する端末にはIPアドレスが割り当てられます。必然的に、端末数の増加に応じて利用されるIPアドレスの数も増えます。そして、ついに2011年、IPv4アドレスの中央在庫が枯渇してしまいました。

IPv4アドレスの中央在庫が枯渇するということは、新しいIPv4アドレスが供給されなくなることを意味します。中央在庫が枯渇しても、まだ供給した先にはIPv4アドレスの在庫がありましたが、それがなくなるのも時間の問題でした。そのため、「IPv4アドレス中央在庫の枯渇」以降、各所でさまざまな形の「IPv4アドレス在庫の枯渇」が発生し、いままで配布できていたIPv4アドレスが供給できず、インターネットの規模が今以上に拡大しにくくなってしまったのです。

このようなIPv4アドレスの在庫枯渇は突然起こったわけではなく、枯渇すること自体、以前から予想されていたことでもあります。実際、枯渇を遅らせる方法として、IPv4アドレスを節約するためのCIDR（Classless Inter-Domain Routing）[†1]、NAT（Network Address Translation）、プライベートIPv4アドレスなど、さまざまな仕組みが考案されました。それと同時に、IPアドレス空間そのものを大きくすべく1990年代に検討が始まったのが、本書の主題であるIPv6です。

IPv4アドレスの在庫枯渇問題に関しては、第18章で詳しく解説します。

1.1　IPはパケット交換技術

インターネットは複数のネットワーク同士をつなぎ合わせて通信できるようにしたネットワークです。その通信は**パケット**と呼ばれる単位で行われます。パケット（packet）は「小包」という意味を持つ英単語に由来しています。

インターネットの前身となったARPANETが誕生したのは1970年頃で、その当時は通信技術としてのパケットそのものが最新の研究対象でした。**パケット交換方式**が実用化されるまで、電話をはじめとする従来の通信で利用されていたのは**回線交換方式**という技術です。

回線交換方式では、通信中に1つの回線を占有します。通信中に回線を占有するということは、実際にデータが流れているかどうかにかかわらず、その回線を他の通信が利用できないということです。回線が占有されている間に他の通信を行いたい場合には、別の回線（通常は物理的な資源）を用意する必要がありました。

一方、パケット交換方式では、データを小分けにして転送することで、あたかも複数のデータが1つの回線を同時に使っているように錯覚させることができます。1つの回線をさまざまな送信元と宛先の通信で共有できれば、回線の利用効率が良くなるので、回線を運用するコストも下がります。回線交換方式より費用対効果が高い、このパケット交換方式を実現する手法のひとつとして研究されていたネットワークが、後に現在のインターネットへと成長しました。

---

†1　RFC 4632：V. Fuller,T. Li, “Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan”, 2006年8月

CC BY NC SA

1.1.1 IPパケットのフォーマット

通信データを小分けにしたパケットは、その名のとおり、データの小包のようなものです。小包を宛先に届けるには、宛先などの情報も必要です。インターネットを流れるすべてのデータは、「IPパケット」と呼ばれる小包によって小分けにされ、それぞれが「IPアドレス」で示される宛先へと届けられます。

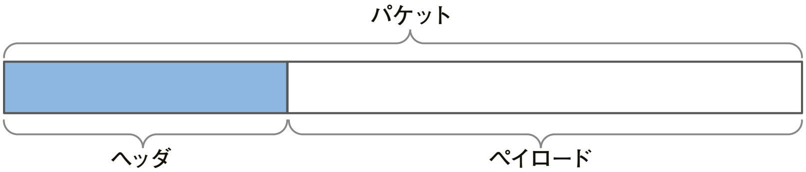
IPパケットのどこにどのように宛先IPアドレスが記載されているのか、通信を行う機器同士の認識がバラバラだと、パケットを正しい宛先に届けることができません。インターネット上でさまざまな機器同士が通信できるのは、「IPパケットがどのような構造をしているか」について、すべての機器が共通の認識を持っているからです。

コンピュータの世界では、データなどを表現する型のことを**フォーマット**（format）と呼びます。「インターネット上を転送されていくIPパケットがどのような構造をしているか」についても、フォーマットとして定義されています。

IPパケットのフォーマットは、大きく**ヘッダ**（header）と**ペイロード**（payload）に分けられます。ヘッダには英語で「見出し」という意味があります。「頭」や「先頭」という意味のheadの末尾にerが付いた英単語であり、文字どおりIPパケットの先頭部分です。もう一方のペイロードは、コンピュータ用語としては「運ばれるデータ」を示します[†2]。IPパケットでは、ヘッダに続くペイロードに転送するデータなどが格納されます。

■ パケットとヘッダフォーマット

図1.1は、ネットワーク上を流れるパケットを図示して説明するときによく用いる、便宜的なパケットの姿です。本書でも、図中でパケットを表すときには、このような絵として描くことがあります。



▶ 図1.1 パケットのヘッダとペイロード

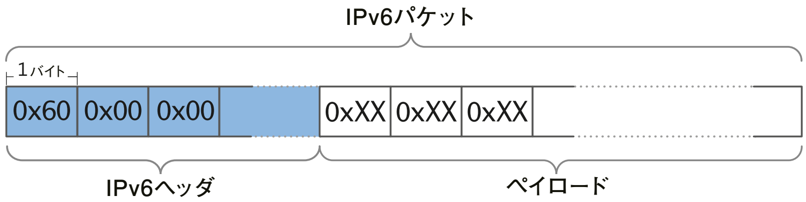
絵では図1.1のように表現されていますが、実際のパケットはバイト列です。たとえば、IPv6パケットを1バイト（8ビット[†3]）ずつ十六進表記で表せば、図1.2のようになります。

IPv6パケットのヘッダには、宛先のIPv6アドレスをはじめとして、そのIPv6パケットをインターネットで転送するために必要な情報が格納されます。これらの情報に注目して、IPv6ヘッダの中身をビットごとに示したのが、図1.3です。このようなヘッダの構成図のことを、一般に**ヘッダフォーマット**と呼びます。

---

†2 運送などにおいて有料で運ぶ貨物を表すpayloadが語源です。payは払うという意味があり、loadは荷物という意味があります。

†3 現在のほとんどのシステムでは1バイトが8ビットですが、過去にはそうでないシステムもありました。そのような事情もあり、ネットワークに関する仕様では、8ビットを「1オクテット」という場合もあります。

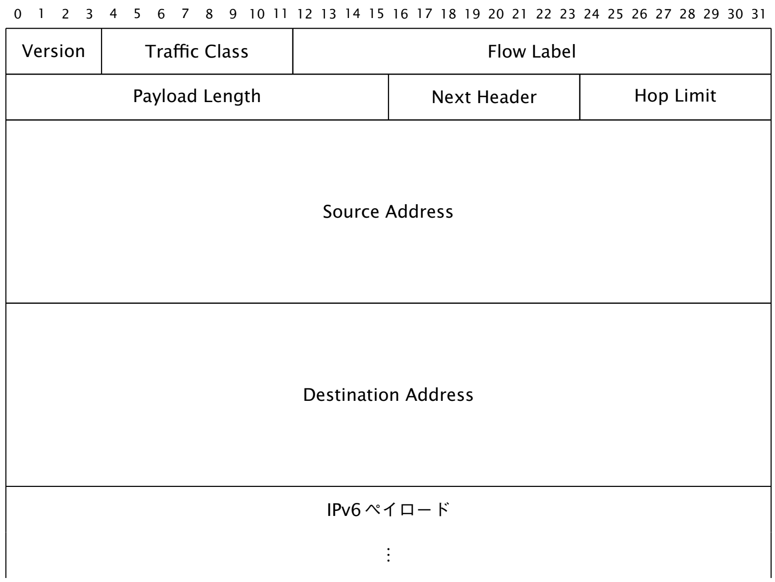


▶ 図1.2　IPv6 パケット

IPv6 パケットのヘッダフォーマットは、インターネットプロトコルを構成する非常に重要な要素です。こうしたフォーマットの各要素のことを**フィールド**と呼びます。IPv6 パケットの各フィールドにどのような情報がどのような形で格納されるかについては、第4章で詳しく説明します。

■ エンディアンとネットワークバイトオーダー

パケットのヘッダフォーマットなど、通信プロトコルやフォーマットを理解するための要素として、エンディアンとネットワークバイトオーダーという概念があります。ヘッダフォーマットの図をなんとなく眺めるのであれば、それらの点をあまり理解しなくても大丈夫であることも多いのですが、実際に通信プログラムを実装したりするときには、これらの概念について理解していないと問題が発生することがあります。ありがちなのは、「リトルエンディアン（後述）を採用しているハードウェアを使って通信プログラムを書いたときに、うまくパケットのヘッダを生成できないバグになってしまう」という問題です。この辺の話は少しわかりにくいかもしれないので、ここではエンディアンとネットワークバイトオーダーについて少し詳しく説明します。



▶ 図1.3　IPv6 パケットのヘッダフォーマット

バイト列であるパケットを、図1.3のようなフィールドからなるデータとして扱うのは、コンピュータです。コンピュータでは、メモリ上に配置したデータをCPUが操作することでさまざまな処理を実行します。そのため、「パケットのバイト列を処理するときメモリ上にどのように配置するか」が問題になります。こうしたバイト列の並べ方を**バイトオーダー**と呼びます。その具体的な種類が**エンディアン**です[†4]。

エンディアンの代表は、**ビッグエンディアン**と**リトルエンディアン**です。その語源は、小説『ガリバー旅行記』の中に出てくる、卵の殻の正しいむき方をめぐる意見の相違から戦争が起きる話にあります。卵の殻を大きい側からむく勢力がビッグエンディアン、小さい側からむく勢力がリトルエンディアンです。コンピュータの世界では、ビッグエンディアンは「大きな桁を表すビット」を含むバイトから最初に並べる方式、リトルエンディアンは「小さな桁を表すビット」を含むバイトから最初に並べる方式を指します。ビット列のうち最も大きな桁を表すビットを**MSB**（Most Significant Bit）、逆に最も小さな桁を表すビットを**LSB**（Least Significant Bit）と呼ぶので、MSB側のバイトから並べるのがビッグエンディアン、LSB側のバイトから並べるのがリトルエンディアンだといえます。

実は、コンピュータがどちらのエンディアンに従うかには統一された基準がありません。つまり、パケットのバイト列をまとめて扱うとき、メモリ上にバイト単位で割り当てられている番地（アドレス）の「小さい側」にバイト列の「上位の桁」を配置するのか、それとも「下位の桁」を配置するのかは、コンピュータによって異なります。

> **NOTE**
>
> RFCなどに示されているヘッダフォーマットは、ネットワーク上で物理的にやり取りされるビットの順番を示しているわけではないことに注意してください。ネットワーク上で物理的にどのような順番でビットが転送されるのかは、物理層を構成する仕組みに依存します。バイトオーダーは、「機器内のメモリ上にどのようにバイトを配置するのか」、あるいは「RFCなどの文書でどのように表現をするのか」という話です。

実際の例で見てみましょう。まず、32ビットの値をビッグエンディアン、リトルエンディアンでバイト単位に並べることを考えます。例として、`0001 0010 0011 0100 0101 0110 0111 1000`という32ビットの値（1バイトずつ16進表記で書くと`0x12 0x34 0x56 0x78`）を、それぞれビッグエンディアンとリトルエンディアンで並べたものを図1.4に示します。リトルエンディアンはLSBが先頭になり、ビッグエンディアンはMSBが先頭になっているのがわかると思います。

注意が必要なのは、コンピュータが複数のビット列を1つの単位として扱うとき、その単位が32ビットに限定されるわけではないという点です。32ビットで扱われるときには、その32ビット内でのMSBとLSBがあり、16ビットで扱われるときには16ビット内でMSBとLSBがあるのです。

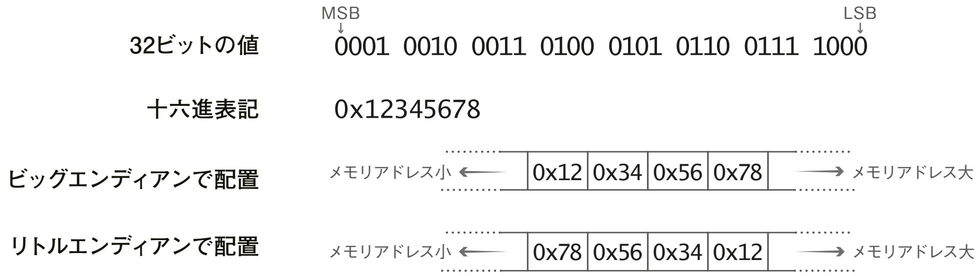
次は、16ビットの値を2つ連続して配置する例です。先ほどの図のように32ビットという単位でMSBとLSBがあるのではなく、16ビットという単位でMSBとLSBがあります。

---

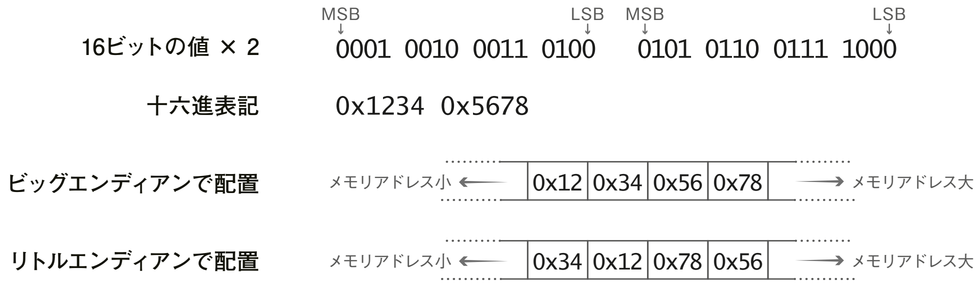
†4 日本語では「エンディアン」ですが、英語では“endianness”と呼ばれているので、RFCなどを読むときには注意が必要です。

CC BY NC SA

図1.5は、0001 0010 0011 0100（1バイトずつ十六進表記すると0x12 0x34）という16ビットの値と、0101 0110 0111 1000（同じく0x56 0x78）という16ビットの値が連続している場合に、これらをビッグエンディアン、リトルエンディアンで並べた例です。



▶ 図1.4　32ビットの値を配置するときのエンディアンの違い



▶ 図1.5　16ビットの値を2つ配置するときのエンディアンの違い

図1.5と図1.4を比べるとわかるように、16ビットの値が2つ並んだ32ビットをリトルエンディアンで配置した場合と、ビット列としては同じ32ビットの値をリトルエンディアンで1つ並べる場合とでは、同じビットの配置になりません。この違いだけを見るとリトルエンディアンのほうに違和感があるかもしれませんが、CPUのレジスタで扱うときの単位が16ビットの値を16ビットずつ独立して扱うので、これでもコンピュータにとって不都合があるわけではありません。実際にバイト列をプログラムで扱うときに便利な面もあるので、システムによってはリトルエンディアンを採用しています。たとえば、インテルのx86系CPUはリトルエンディアンです。

一方、ビッグエンディアンを採用したシステムとしては、かつてのサン・マイクロシステムズ（現在のOracle）によるSPARCなどが知られています。ARMなど、設定によってエンディアンを切り替えられるCPUもあります。

世界中のすべてのコンピュータが同じエンディアンを採用していれば、エンディアンについて気にする必要はないでしょう。しかし、通信プロトコルでは、異なるエンディアンの機器同士でも通信できるようにする必要があるので、エンディアンの扱いについても考慮しなければなりません。異なるエンディアンを採用する機器同士でも問題なく通信できるのは、エンディアンをどうするかを含めて通信プロトコルが定義されているからです。

さて、ここまで説明してきたリトルエンディアンとビッグエンディアンは、「ビット列」の扱

いに関する話です。インターネットで利用するプロトコルという視点で見たとき、8ビットを1単位とした「オクテット単位でのデータの並べ方」という視点もあります。この視点では、8ビットのデータ列がバイトと表現され、その「バイト列」がどのように並ぶのかをバイトオーダーと呼びます。

TCP/IPでも、エンディアンが異なるシステム間でデータを問題なくやり取りできるように、バイトオーダーが事実上ビッグエンディアンに統一されています。インターネットで使われるバイトオーダーでは、MSBを含むオクテット（バイト）を最初に並べることになっています。この、ネットワークで利用されるバイトオーダーのことを、**ネットワークバイトオーダー**と呼びます。それに対し、各コンピュータ内でのエンディアンは**ホストバイトオーダー**と呼ばれます。

NOTE

ネットワークバイトオーダーをTCP/IP全体の仕様として定めるRFCなどの文書はありません。IPv4仕様のRFC 791[†5]のAppendix Bに記載されているバイトオーダーも、「this document（この仕様書）」が対象なので、IPを利用している上位層のプロトコルすべてが同様のルールであると述べているわけではないのです。ただし、各種のヘッダフォーマットをはじめ、厳密にはバイトオーダーが問題になる図は、すべてネットワークバイトオーダーを前提として描かれています。
なお、この並べ方について、IPv4の基本仕様を示しているRFC 791では「英語と同じように（"which they are read in English"）」とされています。インターネットが米国で生まれたからこそ、基本仕様がこのような表現になっているのでしょう。

■ 本書のフォーマット図のバイトオーダー

本書では、パケットヘッダのフォーマット図などを示すとき、IPv4の基本仕様を規定したRFC 791のAppendix Bに示されたものと同様の方式で表現します。RFC 791のAppendix Bには、「このドキュメントで図示したヘッダやデータを転送する順番はオクテットごととする」とあります。したがって、たとえば図1.6のようにバイト列を示した場合、このデータは実際にオクテット単位（8ビット単位）で左側から右側へと転送されるものとします（一番右側に進んだあとは、次の行へと進みます）。

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

1	2	3	4
5	6	7	8
9	10	11	12

▶図1.6　オクテット単位でのデータの転送順序

また、図1.7のようにビット列を表現するときは、一番左側（0番のビット）がMSBを示すものとします。図1.7の8ビットのデータは、0番め、2番め、4番め、6番めのビットが1なので、16進数で示すと0xaaであり、10進数で示すと170です。

†5　RFC 791：J. Postel, "Internet Protocol", 1981年9月

0	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0

▶ 図1.7　オクテット単位でのデータの転送順序

1.1.2 IPパケットを転送するルータ

IPパケットをインターネット上で転送していく機器は**ルータ**と呼ばれます。「経路」という意味を持つroute（ルートもしくはラウトと発音）という英単語から、経路を提供するという意味でrouter（ルータやラウターと発音）というわけです。

インターネットを構成する各ルータは、インターネットの全体像を把握しているわけではありません。ルータの仕事は、IPパケットに記載されている宛先などの情報を見ながら、次のルータ（次ホップ）に向けてパケットを転送するだけです。ルータが「IPパケットのヘッダに記載されている宛先IPアドレスがどこにあるのか」を知っているとは限りません。

ルータは、自分がどのような通信を実現しているのかといった細かいことは気にせず、個々のパケットに記載された宛先IPアドレスに従ってパケットをひたすら転送することだけに専念します。インターネットの設計では、できるだけ単純な作業だけをしているのです。

IPパケットにどのような情報が保持されていて、ルータがそこから何を読み取れるのかは、インターネットプロトコルで決められています。IPパケットを送出する側は、インターネットプロトコルで決められた方式に従ったIPパケットを作成し、送出します。インターネットプロトコルという共通言語があるからこそ、ルータはパケットに記載された情報を読み取りながらパケットを転送できるのです[†6]。

ルータが適切に次の転送先を判断できるのは、「この宛先に届けたいときにはどっちに転送すればいいか」を確認できる**ルーティングテーブル**と呼ばれる情報を持っているためです。

パケットを受け取ったルータでは、IPパケットに記載された「○○宛」という宛先IPアドレス情報を、ルーティングテーブルに記載されている経路情報と突き合わせて、適切な「次ホップ」へと送り出します。宛先IPアドレスとルーティングテーブルとの突き合わせにマッチした「次ホップ」へ転送するだけなので、あまり深く考えなくても処理ができます（図1.8）。

このようにルータは、基本的には「どのようなパケットがいつどこでどのように転送されたのか」をまったく把握せず、各パケットに記載された情報と、あらかじめ用意されたルーティングテーブルの情報だけで稼働しています[†7]。

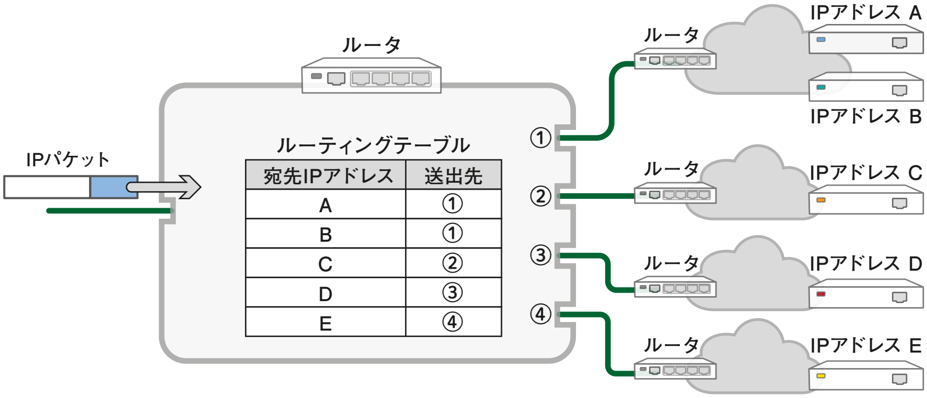
NOTE

ルーティングテーブルの情報がどのように用意されるのかについては、インターネットを構成するルータ以外の要素とネットワークのアドレスについて説明した後、1.1.6項で説明します。

---

†6　インターネットプロトコルは、経路を判断して転送する部分に関しては規定していません。受け取ったIPパケットをどこにどのように転送すればよいのかを判断するための仕組みは、インターネットプロトコルとは別の仕組みとして定義されています。

†7　本書の主題とは異なりますが、1つのルータ内で仮想的に複数のルーティングテーブルを運用するVRF（Virtual Routing and Forwarding）という手法もあります。

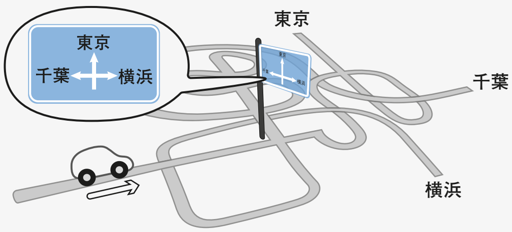


▶ 図 1.8　ルータの中にあるルーティングテーブルの概念図

ルーティングテーブルは道路標識

もう少しわかりやすくするために、現実世界に当てはめて考えてみましょう。いま、宛先が東京都千代田区○×のとある住所、出発地点は名古屋、パケットは目的地へと向かう車とします。車は、名古屋から東京に向かうために、東名高速道路に乗ったとします。

高速道路には行き先を示す道路標識があります。そこには、その車を運転している人が行きたい住所がそのまま書かれているわけではなく、大まかに「東京はこっち」というようなことが書いてあるだけです。東名高速道路を降りて、さらに目的地に近づいていくと、徐々に標識に書いてある地名も詳細なものになっていき、最終的に車が目的地に到着できます。



▶ 図 1.9　ルーティングテーブルは地図ではなく道路標識

ルーティングテーブルは、図 1.9 のように、「○○という宛先へ行くならこっちへ進め」という情報（経路情報）が記されている道路標識のようなものであると考えることができます。

各ルータが把握しているのは、大まかな宛先に対して向かうべき方向を示した経路だけです。ルーティングテーブルには、宛先までの詳細な道筋や、インターネットの全体像は記されていません。車の運転であれば、カーナビや地図を見ながら運転手が宛先までの道を考えて自分で進むこともできますが、パケットの場合は、ルータに設けられた道路標識だけを頼りに、宛先まで場当たり的に向かっていくような感じだと思ってください。ルーティングテーブルは「地図」ではなく、あくまでも「道路標識」なのです。

そして各ルータ単体では、宛先までの方向を示すという単純な作業しかしませんが、複数のルータが互いに連携して転送していくことで最終的にはIPパケットが送信元から宛先へ届きます。そこが、インターネットの面白いところです。

1.1.3 ルータ、ホスト、ノード

少し話の流れが途切れてしまいますが、ここでいくつか用語の定義をしておきます。本書では、ルータ、ホスト（host）、ノード（node）という単語を次のように使い分けています。

ノード（node）という英単語は、節や節点という意味を持ちます。ネットワークを図示したとき、何らかの節点となる存在は、すべて「ノード」です。たとえば、ネットワークの末端に接続された機器も「ノード」ですし、ネットワークの中心部に接続されつつパケットを転送する存在も「ノード」です。先ほどから登場している**ルータ**も、パケットを転送するノードなのです。実際、IPv6の基本仕様を示したRFC 8200[†8]では、「node」という用語の定義を、「IPv6を実装した機器」としています。

ホスト（host）という英単語は、「主人」や「主催者」という意味を持つ英単語です。ユーザに対してコンピュータ資源の利用環境を提供する存在といったニュアンスで、コンピュータそのものをホストと呼ぶことがあります。しかし、ネットワークという文脈でホストという用語が使われるときは、パケットを転送する機能を持たずネットワークの末端に接続されたノードのことを特に示します。RFC 8200では、「ホスト」の定義を、「ルータではないノード」としています。

ネットワークの末端に接続されたノードは、ホストと呼ばれることもありますが、**エンドノード**と呼ばれることもあります。ルータではないノードにはそれより先のネットワークが存在しないので、ネットワークの末端、つまり「エンド」ノードなのです。

1.1.4 IPアドレスの構成

ここまでの説明では、IPパケットはインターネット上でルータによって転送されること、ルータの内部にはルーティングテーブルというものが存在し、そこにはIPパケットの宛先に対応する**経路情報**が記載されていることを紹介しました。では、実際にルーティングテーブルに記載されている情報とは何でしょうか？

ルーティングテーブルに登録されているのは、「IPアドレスの集合体」を表す情報です。つまり、インターネットで経路情報として扱われるのは、機器の個々のインターフェースに付いている具体的なIPアドレスではなく、ある範囲のIPアドレスを集約したもっと大まかな情報です。この情報のことを、**サブネット**とか、単に**ネットワーク**などと呼びます。

11ページのコラムでは、ルーティングテーブルを道路標識のようなものだと説明しました。IPアドレスを「番地情報を含めた詳細な住所」を示す情報だと考えれば、ネットワークは「市町村レベル」を大まかに示す情報に相当します。たとえば、「千葉県浦安市舞浜1番地1」という詳細な住所に対する市町村は「千葉県浦安市」です。道路標識に記載されるのは「浦安市は

[†8] RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017年7月

CC BY NC SA

こちら」といった市町村レベルの情報であり、「千葉県浦安市舞浜1番地1はこちら」のようには示されません。同じように、インターネット上の経路をルータで扱うときには、具体的なIPアドレスではなくネットワークで示します。

IP アドレスは固定長のビット列なので、そのうちの前半部分をネットワークを示すものとして扱い、後半部分を機器のネットワークインターフェースを示すために使います。IPv4 アドレスの場合は、前半部分を**ネットワーク部**、後半部分を**ホスト部**と呼びます。IPv6 アドレスの場合は、前半部分を**サブネットプレフィックス**、後半部分を**インターフェース識別子**と呼びます。いずれの場合も、IP アドレスの全体が異なっていても前半部分が共通であれば「同じネットワーク」と考えます。

NOTE

IPv6 アドレスでは、一部の例外を除いて基本的に、128 ビットのうち前半 64 ビットをサブネットプレフィックス、後半 64 ビットをインターフェース識別子として使います。IPv6 アドレスの構成については 2.4 節で改めて説明します。

一般にコンピュータの世界では、あるデータから特定の情報を取り出すとき、**マスク**（または**ビットマスク**）という処理を使います。口や顔を覆うマスクのように、データの一部分を覆い隠すビット列を使って情報を取り出すわけです。ビット列すべてを使って表したものがIPアドレスであり、その一部分を抜き出したものがネットワークなので、IP アドレスからネットワークを示す部分を取り出すのにもマスクを使えます。そのためのマスクを**ネットマスク**と呼びます。IP アドレス（IPv4 の場合は 32 ビット、IPv6 の場合は 128 ビット）のうち、それぞれの長さのビット列で先頭からネットワークを示すまでの部分をすべて 1、それ以降をすべて 0 にしたものがネットマスクになります。

ネットワークを示す部分だけのビット列を得るには、IP アドレスとネットマスクとでビット演算の AND をとります。ビット演算の AND は、2 つの入力の両方が 1 であるときに 1 になり、どちらかが 0 であれば 0 という結果を返す演算です。IP アドレスとネットマスクを利用して AND のビット演算を行うと、IP アドレスのビットのうち、ネットマスクのビットが 1 となっている部分だけのビットが 1 になります。ネットマスクで 0 となっているビットの部分は「マスクされる」わけです。

IPv6 アドレスとネットマスクの例を見てみましょう。`2001:db8::1` という IPv6 アドレスと、`ffff:ffff:ffff:ffff::` というネットマスクを利用して、サブネットプレフィックスを抽出する場合を考えてみます（図 1.10）。ネットマスクの 128 ビットのうち、上位 64 ビット分が 1 になっているので、IPv6 アドレスの上位 64 ビット以外はすべて 0 になります。そのため、`2001:db8::1` という IPv6 アドレスに対する `ffff:ffff:ffff:ffff::` というネットマスクを使って得られるサブネットプレフィックスは `2001:db8::` になります。

NOTE

128 ビットの IPv6 アドレスは、この例のように 16 ビットずつを「`:`」で区切って十六進表記で書き表します（「`::`」となっている部分は、連続するゼロが省略されていることを示します）。IPv6 アドレスのテキスト表記と省略ルールについては 2.2 節および 2.2.1 項で改めて説明します。

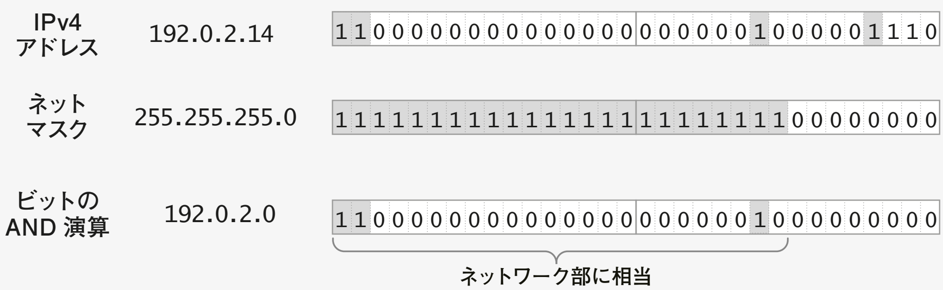
CC BY NC SA



▶ 図 1.10　IPv6 でのネットマスクの例

IPv4 アドレスとネットマスク

IPv4 アドレスとネットマスクの例を見てみましょう。192.0.2.14 という IPv4 アドレスと、255.255.255.0 というネットマスクを利用して、ネットワーク部を抽出する場合を考えます（図 1.11）。IP アドレスとネットマスクとを入力とするビットの AND 演算を行うと、ネットマスクが 255.255.255.0 なので、IP アドレスの 32 ビットのうち最初の 24 ビットの値がそのままとなり、IP アドレスの最後の 8 ビットがすべて 0 になります。そのため、192.0.2.14 という IP アドレスと、255.255.255.0 というネットマスクで表される経路は、192.0.2.0 というネットワークを宛先としたものになります。



▶ 図 1.11　IPv4 でのネットマスクの例

CC BY NC SA

1.1.5 プレフィックス長

ネットワークは、ネットマスクではなく、**プレフィックス長**で表現されることもあります。**プレフィックス**（prefix）は、英語で「接頭辞」や「前に置くもの」という意味を持つ単語です。

プレフィックス長を使ったネットワークの表現では、IPアドレスの前半部分のうち「どこまでのビットで示されるネットワークについて表現するのか」を指定します。ネットマスクを使って同様の表現をすることも可能ですが、その場合には「ネットマスクの先頭から連続する1の数」を表現することになるので、プレフィックス長として示すほうがシンプルです。

プレフィックス長を使ってネットワークを表現するときは、そのネットワークをIPアドレスと「/」に数字が続く形で表現します。たとえば、`2001:db8::/32`という表現は、`2001:db8::`というIPアドレスのうち最初の32ビットによるネットワークプレフィックスである、ということを意味します。

IPv4アドレスを人間にとって見やすい形で書き表すときは8ビットごとにドット区切りにし、IPv6アドレスの場合には16ビットごとにコロンで区切りますが、プレフィックス長は8や16の倍数とは限らないことに注意してください。IPv6では、16ビットごとに`:`で区切った部分の中身を表現しますが、4ビットごとに16進数で表現するため、プレフィックス長が4の倍数となっていると人間にとって理解しやすくなります。

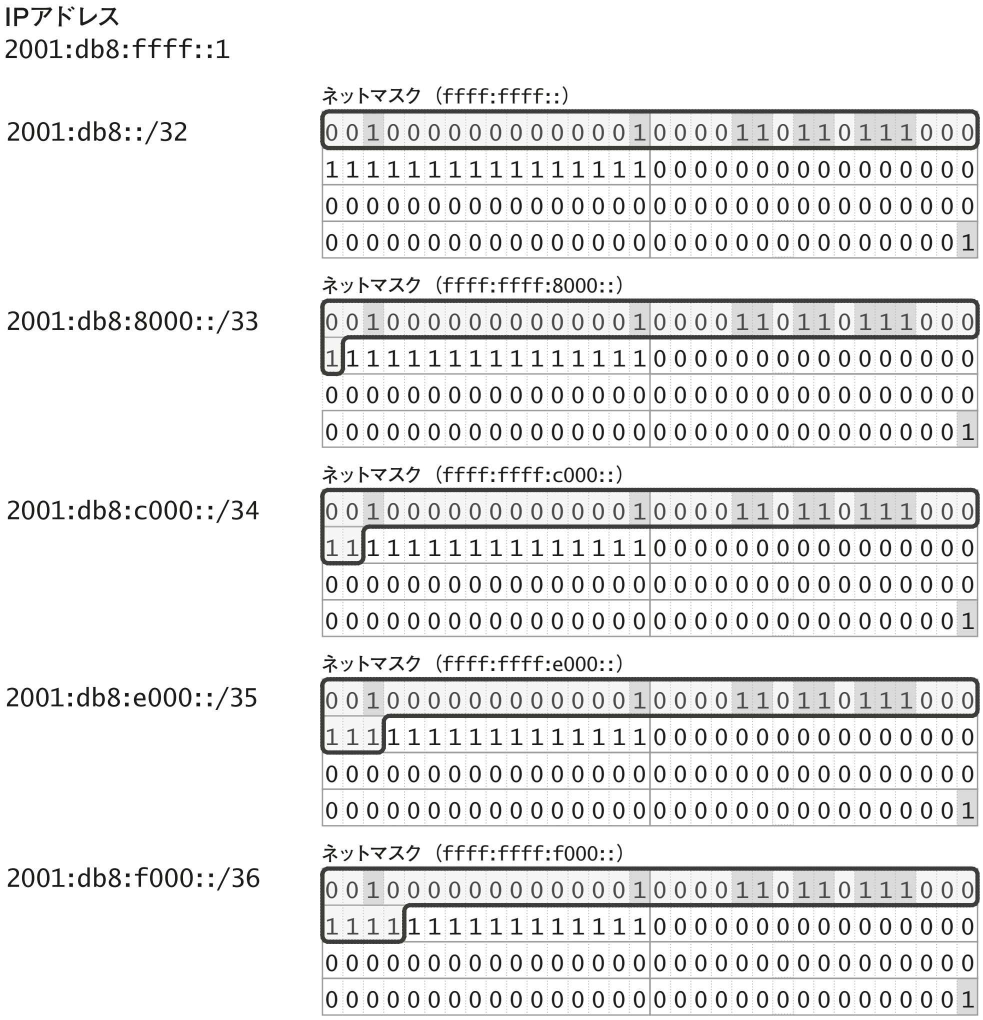
プレフィックス長とネットワークプレフィックスの例をいくつか見てみましょう。図1.12のように、`2001:db8:ffff::1`というIPアドレスがあったとして、プレフィックス長が`/32`、`/33`、`/34`、`/35`、`/36`のとき、それぞれのネットワーク部はどのようになるでしょうか？

プレフィックス長が32の場合、ネットマスクは128ビットのうちの先頭32ビットが1なので、`ffff:ffff::`になります。`2001:db8:ffff::1`と`ffff:ffff::`のAND演算を行うと、`2001:db8::`となるので、`2001:db8::`がネットワークプレフィックスになりますが、このとき、プレフィックス長と一緒に`2001:db8::/32`と表記します。

次は、プレフィックス長が`/33`の場合を考えましょう。プレフィックス長が`/33`の場合、ネットマスクは128ビットのうちの先頭33ビットが1なので、`ffff:ffff:8000::`になります。`2001:db8:ffff::1`と`ffff:ffff:8000::`のAND演算を行うと、`2001:db8:8000::`となるので、`2001:db8:8000::/33`がネットワークプレフィックスになります。このように、プレフィックス長が`/33`の場合は、16の倍数ではないので、`/32`の場合と比べて少しわかりにくくなります。

同様な計算で、`/34`の場合は`2001:db8:c000::/34`、`/35`の場合は`2001:db8:e000::/35`、`/36`の場合は`2001:db8:f000::/36`となります。

IPv4ではエンドノードが接続されているサブネットのプレフィックス長が環境によって異なるのが一般的ですが、IPv6では、エンドノードが接続されている個々のサブネットでは`/64`というプレフィックス長でのサブネットプレフィックスが使われるのが一般的です。その一方で、IPアドレスの割り当てや割り振り、ルーティングなどでは、IPv6でもIPv4同様にさまざまなプレフィックス長が使われます。



▶ 図1.12　IPv6でのプレフィックス長の例（プレフィックス長が32、33、34、35、36の場合）

1.1.6　ルーティングテーブルはどのように生成されているのか

ルータが受け取ったパケットをルーティングテーブルに従って転送する行為は**フォワーディング**と呼ばれています。受け取ったパケットを次ホップへとフォワード（forward）、つまり転送するからです。フォワーディングでは、作成済みのルーティングテーブルを参照しながらパケットを転送します。ここまでの説明も、各ルータがすでにIPアドレスに対応する経路を知っている、つまりルーティングテーブルを持っている状態であることを前提にしていました。

ルーティングテーブルの生成方法にはさまざまなものがあります。どれか1つだけに決まっているわけではありませんが、「人間が手動で設定する静的な方法」と、「インターネット上のルータ同士が協調してルーティングテーブルを自分たちで生成する動的な方法」に大別できます。

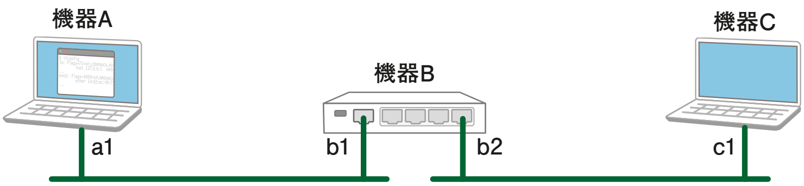
ルータ同士が協調して経路を把握し、動的にルーティングテーブルを生成する仕組みは、インターネットを理解するうえで非常に重要なポイントです。そのため、そのような仕組みはプ

ロトコルとして規定され、利用されています。そのようなプロトコルは**ルーティングプロトコル**と呼ばれています。

以降では、ルーティングテーブルの生成について、まずは静的な方法を説明してから、ルーティングプロトコルによる動的な方法を説明します。

■ 小規模なネットワークでの静的なルーティングテーブル生成例

ルーティングテーブルが作成される過程を知るために、とても単純なネットワークを例にして考えてみましょう。図1.13のような、3台の機器が数珠つなぎになったトポロジがあったとします。**トポロジ**とは、ネットワークの構成やネットワーク全体の形状のことです。



▶ 図1.13　3台の機器が数珠つなぎになったトポロジ

機器Aのネットワークインターフェースはa1、機器Bのネットワークインターフェースはb1とb2、機器Cのネットワークインターフェースはc1とします。

図1.13のようなトポロジでは、複雑なルーティングは必要ありません。末端にいるAとCは、Bへとパケット転送を依頼する以外に方法がありません。間にいるBも、宛先に応じてパケットをAかCに転送すれば済みます。このネットワークでは、間にいるBが「自分のつながっているネットワークをすべて知っている」ので、その情報だけでルーティングが可能です。

図1.13のネットワーク例で、AとBとCに対して静的にルーティングテーブルを作成してみましょう。静的な設定なので、管理者が作成したルーティングテーブルをAとBとCに対して個別に手動で設定します。なお、管理者はこのネットワークのトポロジをあらかじめ把握しているものとします。

まず、Aのルーティングテーブルを考えてみましょう。図1.13のネットワークにはAとBとCしか存在しないので、Aは、BおよびCと通信できれば十分です。Aのa1とBのb1は直接接続されているので、Aが通信に利用しているネットワークインターフェースa1に対してパケットを送信すれば、Bに届くことが期待できます。

また、AがCと通信するためにはBでパケットが転送される必要があります。そこでAのルーティングテーブルでは、AからCへの通信についても、ネットワークインターフェースa1に対してパケットを送信するように手動で設定します。

図1.13の例では、CもAと同様の状況です。そこでCのルーティングテーブルでは、BとAへのパケットの送信について、ネットワークインターフェースc1経由で送信するように手動で設定します。

図1.13のBは、AおよびCと違い、b1およびb2という2つのネットワークインターフェースを持っています。まず、BがAと通信したい場合について考えましょう。BはAとb1を通じて接続しているので、BがAにパケットを送信したいのであれば、b1へとパケットを送信する

CC BY NC SA

必要があります。そのため、Bのルーティングテーブルには、「Aへの経路はb1」という設定を手動で施します。

同様に、「Cへの経路はb2」というルーティングテーブルをBに追加したうえで、Bがルータとして動作するようにすれば（つまり受け取ったパケットを他のネットワークインターフェースへと転送できるように設定すれば）、Bを経由してAからのパケットがCへと転送できるようになります。

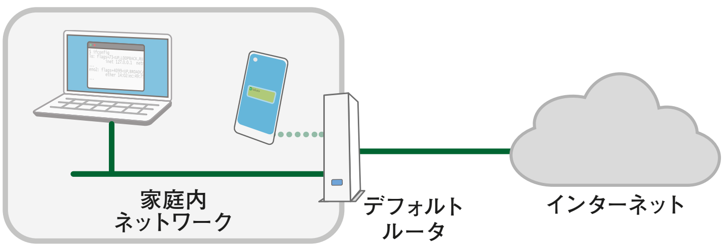
このように、A、B、Cすべての機器を個別に手動設定することで、図1.13のトポロジで機器同士が相互に通信できるようになります。

■ デフォルトルータ

先ほどの例では、機器AとCはネットワークインターフェースを1つしか持っていないので、「どのような通信であれ、とりあえず自分のネットワークインターフェース経由でパケットを送信する」という設定でも実は通信できてしまいます。どうせ出口は1つしかないので、深く考えずにすべてのパケットをその1つのネットワークインターフェースでやり取りすればいいからです。このような状況では、通常は**デフォルトルータ**を設定します[†9]。

コンピュータ用語としてのデフォルト（default）という単語には、適切な選択肢が存在しない場合に採用される標準の設定という意味があります。つまり、デフォルトルータは、明示的に指定された経路以外の宛先へのパケットを送信してもらうためのルータです。明示的に指定されていない「その他全部」がデフォルトになります。この場合には、そもそも「その他全部」ではない経路が1つも設定されていないので、毎回必ず「その他」の設定が採用されます。

皆さんが普段、家庭や職場のネットワークで利用する手もとの機器では、デフォルトルータが設定されていることが多いでしょう。ルータ1台に依存すればよい機器は、依存すべきルータを「その他全部」として設定するという単純な設定により、インターネットと通信ができるようになるからです。たとえば、図1.14のような家庭内ネットワークとインターネットの橋渡しをするルータが1台だけという単純な状況を考えてみます。



▶ 図1.14　家庭内ネットワークで利用されるデフォルトルータ

図1.14のような環境では、家庭内ネットワークに接続する機器は「インターネットと通信をするのであれば、このルータにパケット転送を依頼すればOK！」という設定で十分でしょう。このような場合、機器にデフォルトルータを設定するだけでインターネットとの通信が可能になります。

†9　**デフォルトゲートウェイ**という表現もありますが、本書ではデフォルトルータという表現を使います。

CC BY NC SA

■ 動的なルーティングテーブル生成

では、デフォルトルータの先にあるルータでは、どのようにしてルーティングテーブルを用意しているのでしょうか？ すべての相手までの経路情報をあらかじめ手動で設定することもできなくはないでしょうが、ネットワークに何らかの変化が発生するたびに設定を手動ですべて変更しなくてはなりません。ネットワークの規模が大きくなれば、すべてを手動で設定するのは困難です。小規模ではないネットワークでは、ルータ同士が協調しつつ経路情報をやり取りして自動的にルーティングテーブルを計算する動的ルーティング（ダイナミックルーティング）が使われます。そのための仕組みが**ルーティングプロトコル**です。

動的ルーティングにもいくつかの種類があります。なかでもシンプルなのは、すべてのルータが自分の隣にある機器を伝言ゲームのように伝えていく、ディスタンスベクタ型と呼ばれる方式です。ディスタンスベクタ型のルーティングプロトコルとしては、**RIP**（Routing Information Protocol）が有名です。ただし、RIPそのものはARPANET時代から使われ続けている非常に古いルーティングプロトコルであり、IPv4でしか使えません。IPv6用のRIPとしては、オリジナルのRIPが改良されたRIPv2のIPv6拡張版として、1997年にRFC 2080[†10]とRFC 2081[†11]で**RIPng**というプロトコルが規定されています。RIPngは、1990年代後半から2000年代前半ぐらいまで、IPv6のためのルーティングプロトコルの実装がほかに存在しない状況だったこともあり、比較的よく使われていたプロトコルです。現在も小規模ネットワークで動的ルーティングを利用する選択肢となりえますが、昔ほど利用頻度は多くないでしょう。

すべてのルータが「自分の周囲にある機器の一覧表」を他のすべてのルータと共有し、それぞれが集めた一覧表からネットワーク全体の地図を作って、それをもとに各ルータでルーティングテーブルを作るという方法もあります。この方法のルーティングプロトコルはリンクステート型と呼ばれ、その代表は、組織やプロバイダの内部ネットワークでよく利用される**OSPF**（Open Shortest Path First）です。IPv4で利用されているOSPFは、OSPFv2というバージョンで、「自分の周囲にある機器の一覧表」をサブネット単位で計算します。IPv6では、1.2節で説明するリンク単位で計算を実行する**OSPFv3**が利用されます。これは、次章以降で後述するようにIPv6では同一リンク上に複数のサブネットが存在可能であり、かつ同一リンク上の異なるサブネット同士が直接通信することが可能だからです。OSPFv3は、RFC 5340[†12]で規定されています。OSPFv3について詳しくはRFC 5340を参照してください。

NOTE

OSPFv3が作られた当初は、OSPFv3がIPv6専用のプロトコル、OSPFv2がIPv4専用のプロトコルとなっており、IPv4とIPv6の両方でOSPFを使うにはOSPFv2とOSPFv3を両方使う必要がありました。その後、RFC 5838[†13]によってOSPFv3が複数アドレスファミリに対応し、IPv4とIPv6の両方を扱えるようになっています。

---

†10 RFC 2080：G. Malkin, R. Minnear, “RIPng for IPv6”, 1997年1月
†11 RFC 2081：G. Malkin, “RIPng Protocol Applicability Statement”, 1997年1月
†12 RFC 5340：R. Coltun, D. Ferguson, J. Moy, A. Lindem, “OSPF for IPv6”, 2008年7月
†13 RFC 5838：A. Lindem, S. Mirtorabi, A. Roy, M. Barnes, R. Aggarwal, “Support of Address Families in OSPFv3”, 2010年4月

CC BY NC SA

OSPFと同じリンクステート型のルーティングプロトコルとして、**IS-IS**（Intermediate System to Intermediate System）と呼ばれるものもあります。IS-ISは、IPv4での利用を前提としたルーティングプロトコルではなかったこともあり、IPv6拡張を規定したRFC 5308[†14]は非常にシンプルな内容となっています。

OSPFやIS-ISの仕組みは、「全員のお隣さん情報がわかればネットワークの全体像がわかるよね」という発想を背景にしています。そのため、話としては単純ですが、インターネット全体のような規模で利用するのは現実的ではありません。ネットワークの規模が大きくなってしまうと、「自分の周囲にある機器の一覧表」をやり取りするための通信量が膨大になるだけでなく、どのネットワークがどのようにつながっているのかを各ルータで計算するのにかかる計算量も膨れ上がってしまい、使い物にならないからです。したがって、インターネット全体で使うためのルーティングの仕組みと、組織内などで使うためのルーティングプロトコルとでは、そもそも別の発想が必要になってきます。

冒頭でも述べたように、インターネット全体は「ネットワークのネットワーク」です。そのため、ルーティングプロトコルにも、各組織が自分のネットワーク内のルーティングに責任を持ち、ネットワーク同士が協調して「ネットワークのネットワーク」でのルーティングを行う仕組みが必要になります。現在のインターネットでは、各組織が責任をもってルーティングを行うネットワークのことを**AS**（Autonomous System、**自律システム**）と呼んでおり、AS同士をつなぐのに**BGP**（Border Gateway Protocol）と呼ばれる特別なルーティングプロトコルが利用されています。BGPが、多くの機器が含まれている「ネットワーク」をASという単位にまとめ、AS間での経路制御を実現しているといえるでしょう。

現在、AS間でIPv4の経路情報を扱っているのは、BGP4というバージョンです。このBGP4をマルチプロトコル対応に拡張し、IPv6のルーティングも行えるようにしたものは**BGP4+**と呼ばれ、RFC 4760[†15]で定義されています。

NOTE

BGP4+によりIPv6に関するルーティング情報の交換も可能になりましたが、**ルータID**という識別子ではIPv4アドレスが必要です。また、BGP4ルータ同士でやり取りするときに使うIPプロトコルは、BGPでやり取りする経路情報とは独立なので、IPv4とIPv6のどちらでも使えます。

---

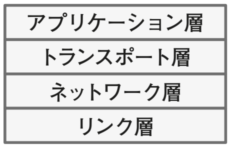
†14　RFC 5308：C. Hopps, “Routing IPv6 with IS-IS”, 2008年10月
†15　RFC 4760：T. Bates, R. Chandra, D. Katz, Y. Rekhter, “Multiprotocol Extensions for BGP-4”, 2007年1月

CC BY NC SA

1.2 層に分かれるネットワーク

1.1 節では、インターネットを支えるプロトコルである IP について説明しました。しかし、実際のインターネットは IP が提供する仕組みだけでは実現できません。

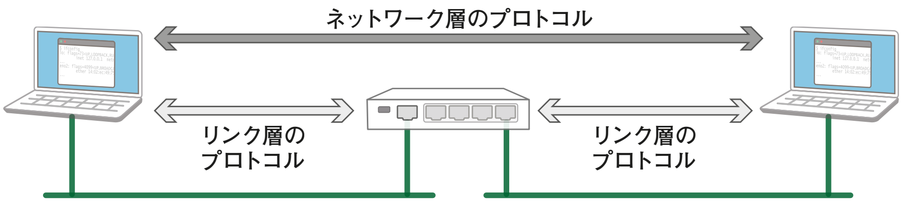
インターネットを理解するための常套手段は、さまざまな機能を「層に分けて考える」というものです。現在のインターネットを支える機能を層に分けると、図 1.15 のような 4 つの階層になると考えられます[†16]。



▶ 図 1.15 インターネット階層モデル

リンク層で扱うのは、隣の機器との接続性です。隣の機器と物理的にケーブルなどでつながっているかどうかだけでなく、論理的につながっているかどうかが問題です。たとえば、光ファイバで 2 台の機器を接続しても、それだけでは通信できません。光ファイバでつながっている機器との通信方法を決める必要があります。そのような、隣の機器との通信方法までを含めて扱うのがリンク層です。

リンク層の上位の層は、**ネットワーク層**と呼ばれます。この層では、リンク層が扱う「隣」よりさらに離れた機器との通信を実現します。図 1.16 のように、リンク層が実現した隣との接続からデータを得たうえで、それをさらに隣へと渡すのがネットワーク層の仕事です。



▶ 図 1.16 リンク層とネットワーク層の違い

ネットワーク層では、隣の機器だけでなく、離れた地点との通信も扱います。これは、ここまでの説明で何度も出てきたルータの仕事だといえます。つまり、IPv4 と IPv6 はネットワーク層のプロトコルです。すごく乱暴に要約すると、インターネットではルータを越えないのがリンク層、ルータを越えるのがネットワーク層の通信だといえます。

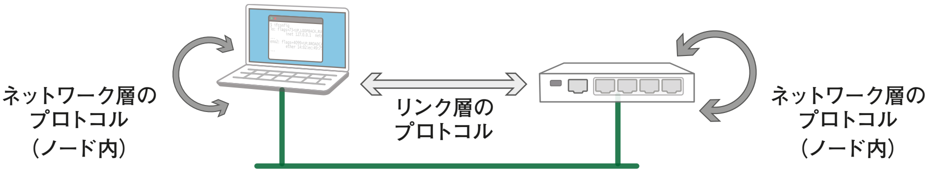
NOTE

リンク層で使われるプロトコルによって通信できる範囲を「リンク」と表現します。たとえば、リンク層のプロトコルの一種であるイーサネットであれば、(フィルタなどの特殊な設定がない状態で)イーサネットフレームが直接届く範囲が「リンク」です。また、そのような状況において「イーサネットフレームが届く範囲に存在していること」を「同一リンクにある」と表

---

†16 RFC 1122 では、リンク層、IP 層、トランスポート層という分け方をしています。本書では、RFC 1122 で IP 層と表現している部分をネットワーク層と表現しました。

現します。
本文で説明しているように、インターネットで同一リンクにあるノード間ではルータを越えずに通信が可能です。ただし、その場合でも通信時にはノード内でネットワーク層の機能は介在しているので注意してください（図1.17）。



▶ 図1.17　同一リンクの通信におけるルータの役割

インターネットが、このような階層に分けて設計されているのには理由があります。もしインターネットに階層構造がなかったら、複数の物理回線にまたがるような遠隔地との間で通信しようとする場合、お互いがネットワークに関するすべての情報を把握する必要があるでしょう。たとえば、「通信相手までの経路には、物理回線として銅線と光ファイバと無線が使われている」といった情報を、通信を開始する前に末端の機器がすべて把握している必要があるかもしれません。これではあまりにも不便です。そこで、物理回線の特性に左右されるような「すぐ隣の機器との通信」と、「離れた地点との通信」とを分けてプロトコルを定義する、階層構造の設計がとられているのです。

階層構造の設計のおかげで、各ルータが接続している物理的な回線の特性や、そうした特性に左右される「すぐ隣の機器と通信」の実現方法に煩わされることなく、ネットワークを越えた通信をするというインターネットプロトコルの仕組みが作れます。IPv4とIPv6という2つのインターネットプロトコルが、いずれも同じ回線の上で動作できるのは、この階層構造の設計の恩恵だといえるでしょう。

ネットワーク層プロトコルであるIPv4やIPv6で決められているのは、通信相手にIPパケットを届けるまでの仕事です。宛先にどのようなIPパケットをどのように届けるか、あるいは、届いたIPパケットをどう扱うかといった仕事は、ネットワーク層の上位の**トランスポート層**の役割になります。パケットを届ける通信そのものと、そのパケットの扱い方が、別の層の機能として設計されているのです。

トランスポート層を担うプロトコルとしては、どのようにIPパケットを扱ってほしいかに応じて、複数の選択肢を選べます。現在、IPパケットを運ぶトランスポート層のプロトコルとして広く利用されているのは、TCP（Transmission Control Protocol）とUDP（User Datagram Protocol）です。ただし、この2つだけがトランスポート層の役割を実現するプロトコルというわけではなく、ほかにもさまざまなプロトコルがあります。

CC BY NC SA

OSI参照モデル

層に分けるやり方としては、歴史的に7つの層に分ける**OSI参照モデル**がよく利用されていました（図1.18）。

7	アプリケーション層
6	プレゼンテーション層
5	セッション層
4	トランスポート層
3	ネットワーク層
2	データリンク層
1	物理層

▶ 図1.18　OSI参照モデル

OSI参照モデルを現在のインターネットにそのまま直接当てはめるのは難しいとも言えそうですが、歴史的経緯で今でも説明に使われることがあります。また、第5層と第6層を除く層の番号は、今でも使われることがあります。

OSI参照モデルでは、イーサネットなどのリンク層が担う役割は第2層に、IPv4とIPv6が担うネットワーク層の機能は第3層に割り当てられます。そのため、それぞれの層（レイヤー）について言及するとき、**レイヤー2**、**レイヤー3**という表現を使うことがよくあります。同様に、レイヤー2のスイッチングを行う装置を**L2スイッチ**、ルータのことを**L3スイッチ**と呼ぶこともあります。また、それより上位のアプリケーションプロトコルのデータまで見てパケットの処理をする機器を**L7スイッチ**などと呼ぶこともあります。

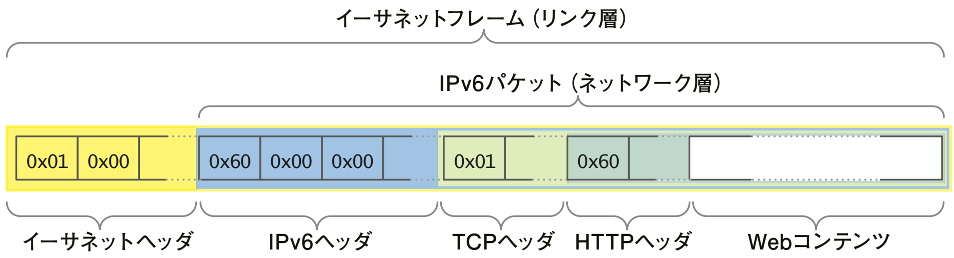
1.2.1　階層構造とパケット

プロトコルを「層」で理解することを念頭に置きながら、インターネット上でやり取りされるパケットの流れを見てみましょう。リンク層のプロトコルであるイーサネット上で、IPv6を使い、Webのトラフィックを運ぶ例を考えてみます（図1.19）。このとき、物理的な回線を流れる信号の正体は、WebのトラフィックのデータにTCPのヘッダが付き、それにIPv6のヘッダが付き、それにイーサネットヘッダが付いたイーサネットフレームと呼ばれるデータです。IPパケットのヘッダについてはすでに説明しましたが、同じように階層ごとにそれぞれヘッダがあり、そのヘッダの直後に次の階層のプロトコルのヘッダが続いていることがわかります。

ヘッダの次に別のプロトコルのヘッダが続くというのは、パケットの中に別のパケットが入っているようなものです。たとえば、図1.19の例でIPv6パケットが運んでいるデータ部分（ペイロード）は、TCPというトランスポート層のプロトコルのヘッダとデータです。パケット通信におけるデータは、ロシアのマトリョーシカ人形のように、入れ子構造になっているのです。

IPv6ペイロードの先頭部分には、上位層であるトランスポート層のプロトコルのヘッダのほかに、IPv6拡張ヘッダやICMPv6ヘッダが続く場合があります。IPv6パケットがペイロードとして運んでいるデータの先頭部分がどのようなヘッダであるかは、IPv6ヘッダにあるNext

CC BY NC SA



▶図1.19　IPv6でのWebトラフィックを運ぶイーサネットフレームの例

Headerというフィールドで示されます。ペイロードの先頭が図1.19のようにTCPの場合、IPv6ヘッダのNext Headerフィールドの値は、TCPを表す「6」になります。

なお、IPv4パケットの場合にはヘッダにProtocolというフィールドがあり、TCPを運ぶパケットの場合は、このProtocolフィールドの値が6になります。一見するとIPv4とIPv6とでフィールドの名前が違うだけにも思えますが、この違いはIPv4とIPv6の本質的な違いでもあります。この違いに関しては、4.3節で解説します。

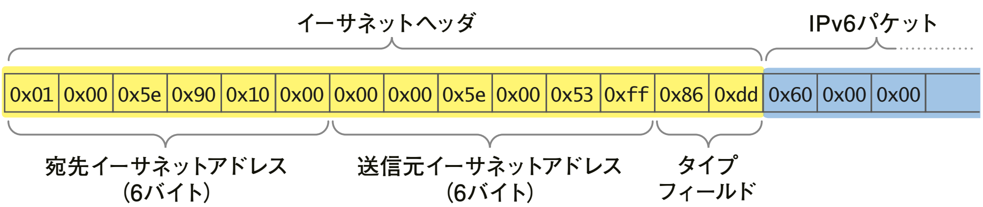
NOTE

IPv6ヘッダのNext Headerフィールドが「No Next Header」の場合には、ペイロードの先頭部分に別のヘッダが続きません。これについては4.3.5項で説明します。

1.2.2　イーサネットにおけるIPv4とIPv6の区別

リンク層のプロトコルで、IPv4とIPv6を区別できるようになっている場合もあります。たとえば、リンク層のプロトコルとして一般的なイーサネットでは、IPv6とIPv4とで、イーサネットフレームの中のタイプフィールドに指定する値が異なります。

イーサネットにおけるIPv6パケットの扱いは、RFC 2464[†17]で規定されています。RFC 2464によれば、IPv6パケットを運ぶイーサネットフレームのタイプフィールドは0x86ddとなります（図1.20）。



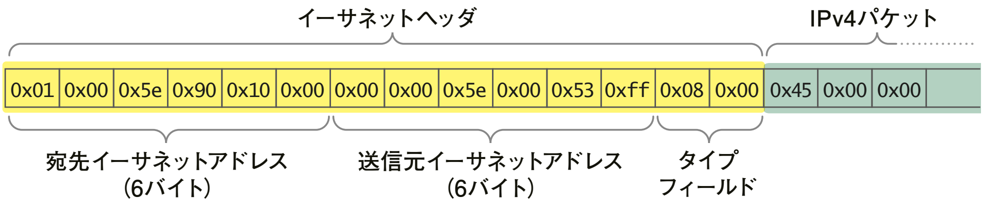
▶図1.20　IPv6パケットを運ぶイーサネットフレーム

一方、IPv4パケットを運ぶイーサネットフレームのタイプフィールドは、0x0800になります（図1.21）。IPv4とIPv6が、リンク層プロトコルであるイーサネットでまったく異なるパ

†17　RFC 2464：M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks", 1998年12月

CC BY NC SA

ケットとして扱われることがわかります。



▶ 図1.21 IPv4パケットを運ぶイーサネットフレーム

イーサネットにおいてIPv4とIPv6とで扱いが異なる要素はほかにもあります。それは、イーサネット上でのマルチキャストで利用されるアドレスです。IPv4では`01:00:5e:XX:XX:XX`が利用されますが、IPv6では`33:33:XX:XX:XX:XX`が利用されます。なお、IPv6マルチキャストのためのイーサネットアドレスとして先頭2オクテットが`33:33`のものが利用されることは、RFC 2464のSection 7で示されています。

1.3 トランスポート層の役割

インターネットの通信で大事なのは、送り出したIPパケットそのものではなく、IPパケットが運んでいるデータのほうです。もし送り出したIPパケットが確実に転送されず、喪失したり変質したりして相手に正しくすべてのデータが届かなかったなら、同じデータを別のIPパケットに入れて再度送り直せばよいのです。そのためインターネットは、「パケットが届くかどうかをネットワークそのものでは保証しない」という、いい加減ともいえる大雑把な仕組みになっています。

もし、送信するデータをすべて確実に相手に届けたい場合は、そのための処理を末端の機器に任せます。途中の経路ではパケットが失われるかもしれないけれど、最終的にはすべてのデータが相手に届くように、末端の機器同士ががんばって失われたパケットを再度送り直すのです。経路の途中にあるルータは個別の通信には関与せず、個別の通信を行っている当事者同士が必要な場合には各自で通信の状況を把握するというのが、インターネットの本来の設計です。

このような設計が可能なのは、1.2節で説明した階層構造のおかげです。IPパケットを届ける部分はネットワーク層で担いますが、この段階ではパケットの到達は保証せず、パケットの到達性などは上位のトランスポート層で保証します。末端の機器でがんばる仕組みを実現している代表的なトランスポート層のプロトコルがTCP（Transmission Control Protocol）です。

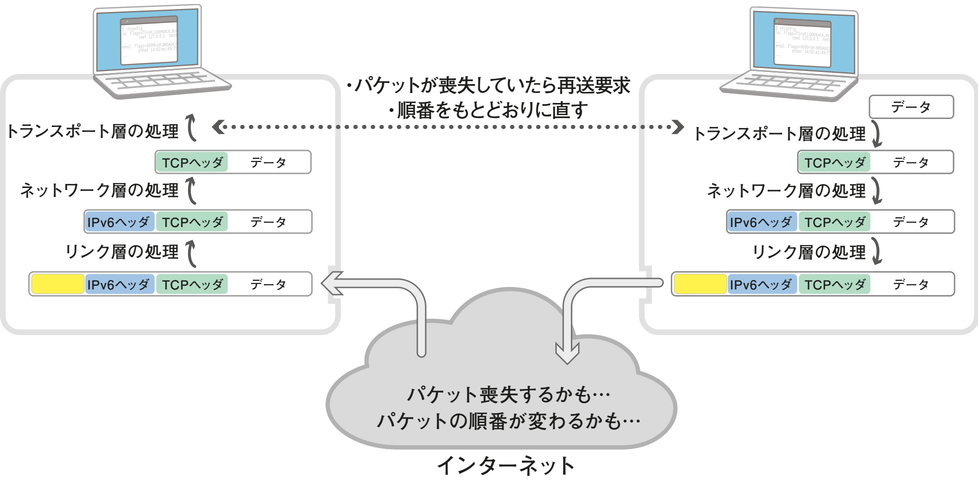
TCPは、IPv4であってもIPv6であっても同じように使えます。ネットワーク層のプロトコルが異なっても、トランスポート層のプロトコルとしてTCPを同じように使えるのは、インターネットの階層構造による恩恵だといえるでしょう。

NOTE

IPパケットは送信元から送り出されて宛先までルータを転送されていくだけなので、IPv6やIPv4によって通信の双方が接続されるわけではありません。しかし、TCPを使うことで、イン

ターネットを介して通信する双方の間に「仮想的な接続」が存在しているかのような状態になります。TCPによって、離れた相手とあたかも直接つながっているかのような錯覚を起こさせるのです。この仮想的な接続は**バーチャルサーキット**と呼ばれています。

TCPには、図1.22のように、相手に正しく届かなかったと推測されるパケットを再度送信する機能があります。TCPの処理で相手に正しく届いていないパケットがあると判明すると、TCPは届かなかったパケットが運んでいたデータを再度別のパケットで送信し、データの到達性を保証します[†18]。



▶ 図1.22　末端の機器同士がパケット到達性を確保

経路の途中で喪失したパケットがないかを検出するために、TCPでは送信側と受信側で「受信完了パケット」をやり取りします。具体的には、受信側が送信側に対して「ここまでのパケットは受け取り終わったよ」という通知を返します。この通知は**ACK**（アックと読みます）と呼ばれています。ACKは、「確認」という意味を持つ「Acknowledgement」という英単語を略したものです。送信側では、パケットを送信したあとに受信者側からのACKが届かないと、「受信側にパケットが届かなかった」と判断します。そして、同じデータを含むパケットを送信し直します。

TCPには、送信側で送られたのと同じ順番でデータを受け取るための機能もあります。インターネットで転送されるパケットは、途中で破棄されるだけではなく、経路上で順番が変わってしまうこともあるからです。たとえば、「か」と「ば」を順番に別々のパケットで送ったとき、送信されたときと同じ順番で受信者側に届けば「かば」ですが、パケットの順番が変わったままで解釈してしまうと「ばか」になってしまいます。順番が変わってしまうとデータの意味も変わってしまいます。インターネットを介したやり取りでデータの順番が変わってしまわないようにすることも、TCPの大事な役目なのです。

TCPは、インターネットを構成する非常に重要な要素です。インターネットに関連するプロトコル群（インターネットプロトコルスイート）をまとめて語るときに**TCP/IP**と表現するこ

[†18] TCPにはさまざまな種類があり、種類によって実際の動作は異なります。

とからも、TCPがインターネットの根幹を成していることがわかります。実際、Webや電子メールといったインターネットにおける通信アプリケーションの大半は、TCPを前提にして設計されています。TCPを利用するアプリケーションは、TCPが裏でがんばることによって、あまり深く考えずに離れた相手と確実にデータをやり取りできるのです。

ここで重要なのは、この「あまり深く考えずに」という点です。アプリケーションを作る人が、経路の途中でパケットが喪失したり到着順が前後したりする可能性を意識せずにプログラムを書けるというのは、ネットワークの普及にとってとても大事なポイントです。インターネットを利用するすべての人が、ネットワークそのものの挙動を細かく知りつつ通信しなければならないような環境では、適切に動作するアプリケーションを作れる人は限られてしまいます。もしTCPが存在しなかったら、インターネットはここまで広く普及しなかったかもしれません。

NOTE

ネットワーク層であるIPの処理に必要な情報がIPヘッダに格納されたように、トランスポート層であるTCPの処理に必要な情報はTCPヘッダに格納されます（図1.19も参照）。TCPヘッダの各フィールドや、それらの情報をどのように使ってTCPの機能が実現されているかについては、本書では説明しません。他の書籍や文献などを参考にしてください。

インターネットにおける通信の大半はTCPによるものですが、音声や動画の通信のようにリアルタイム性が要求されたり、マルチキャストやブロードキャストによって同時に多数の相手と通信したいのでTCPでは都合が悪かったり、TCPほどの機能が不要な場合もあります。そのようなアプリケーションで使われるトランスポート層のプロトコルとして、UDP（User Datagram Protocol）というものもあります。TCPと比べると、UDPはアプリケーションプログラマが自分で必要な機能を自作するカスタム設定用の通信プロトコルであるという見方もできます。

また、携帯電話の規格であるLTE（Long Term Evolution）などでも使われるSCTP（Stream Control Transmission Protocol）[†19]や、QUICという新たなトランスポート層プロトコルもあります[†20]。QUICは、コネクション確立時のハンドシェイクや、暗号化の初期設定などにかかる遅延を軽減するなど、さまざまな工夫によって、低遅延な通信を実現することを目指しています。QUICの当初のユースケースは、現時点ではTCPの利用が前提となっているWebトラフィックを低遅延で扱うことですが、他の用途でも使えることが期待されています。

インターネットの設計思想に見られる分散処理

パケットの到達性などを末端の機器同士でがんばる設計の最大の特徴は、分散処理になることです。もしルータでがんばる設計だとしたら、転送するすべての通信についてルータ自身でもその詳細を把握する必要があります。末端の機器でがんばる設計なら、各機器が自分たちの通信について把握すれば十分であり、必要な処理を分散できます。

---

†19 SCTPはRFC 4960参照。

†20 RFC 9000：J. Iyengar, M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport”, 2021年5月

CC BY NC SA

このような分散処理は、インターネットの設計思想において随所で見られる重要な概念です。

分散処理という特徴は、パケットの到達性だけでなく、インターネットの運用形態にも見られます。インターネットという1つの巨大なネットワークは、たくさんの管理主体がそれぞれ自律分散的に協調しながら運用され続けているからです。

現在のインターネットにおいて組織や国境を越えた分散管理が可能になっているのは、インターネットを支えるさまざまなプロトコルの仕様がオープンであるという特徴によるところが大きいといえるでしょう。仕様がオープンであるという特徴については1.4節で詳しく説明します。

1.4　オープンなプロトコルとRFC

本章の冒頭ではプロトコルのことを「どのようなデータをどのような方法で送受信するのかの取り決め」と説明しました。プロトコルはよく「言語」にたとえられます。たとえば、日本語と英語では話が通じませんが、双方が日本語を使う、もしくは双方が英語を使うなど、同じ言語でそろえれば話が通じます。言い換えると、共通の言語があれば話は通じます。言語の例だと、語彙や文法をそろえるという観点しか説明できませんが、言語を使った実際のやり取りでは、「どちらから話しかけるか」や「どういう手段でやり取りするか」といった点についても双方の了解が必要です。そういった点を含めて、通信を行うための取り決め（約束事）を規定したものがプロトコルです。

ここでは、インターネットを構成しているプロトコルがどのように規定されているのかについて説明します。

1.4.1　インターネットのプロトコルはオープン

インターネットの通信仕様を規定している基本的なプロトコルの仕様はオープンです。つまり、誰にでも無償で公開されています。言い換えれば、誰もが無料で、インターネットを構成している基本的なプロトコルの内容を知ることができます。どこの国の誰でも、どんな企業でも、インターネットに接続できる機器を作れるということです。そして、同じプロトコルを使ってさえいれば、どんな機器でも、たとえ個人が趣味で作った機器でも、つながってしまいます。「誰でも機器を作れる」ということは、世界中で分散してネットワークの構築や管理ができるということでもあります。

インターネットのプロトコルがオープンである、つまり誰にでも公開されていることは、当たり前のことではありません。実際、インターネットの外に目を向けると、世界はオープンではないクローズドなプロトコルやデータのフォーマット形式であふれています。

例として、プロトコルとはちょっと違いますが、データのフォーマット形式について考えてみましょう。日々接している映像や音声といったデータのフォーマット形式も、プロトコルと同様に、どこかで規定された約束事にみんなが従うことが重要になります。しかし、こうしたデータのフォーマット形式にはクローズドなものが少なくありません。よく利用されている動画のフォーマットのなかには、フォーマットについて知るために、規定を定めているコンソー

CC BY NC SA

シアムに法人として加入して仕様が記載された資料を購入する必要がある場合もあります。特定の企業のクローズドなフォーマットでしか再生できないような映像や音声のプレイヤーさえあります。

そのような世界では、個人が趣味で情報を調べることはもちろん、企業などで仕様に対応する製品を作ることも簡単ではありません。誰もが自由に動画や音楽を再生する機器を作れないので、その形式でしか流通していない動画や音楽はごく限られた範囲内でしか広まらないでしょう。

これに対し、インターネットを利用した通信を行うための仕様はオープンであり、それが世界中の共通認識となっています。結果として、誰もが機器を作ってインターネットに接続できます。プロトコルがオープンであることは、時として見落とされがちですがとても大切なことであり、インターネットの非常に大きな特長なのです。

1.4.2 オープンな仕様としてのRFC

TCP/IPに関連する仕様は、主に**IETF**（Internet Engineering Task Force）という組織で議論されています。IETFで議論された文書は、**RFC**（Request For Comments）という形で公開されています。IPv6の仕様も、RFCとして標準化されています。インターネット上の通信仕様を記載した文章がRFCであり、RFCを作るための公開された議論を行う場所がIETFということです。

IETFの存在は、インターネット全体にとって非常に大きな意味を持っています。インターネットが運用的にも文化的にもいまの姿のインターネットになったのは、RFCという形で誰もが読める仕様書が存在することと、そのRFCを作成するためのIETFという公開された組織の存在が非常に大きかったといえるでしょう。

なお、通信に関する規格を標準化している国際的な機関は、IETFだけではありません。**ITU**（International Telecommunication Union、国際電気通信連合）や**ISO**（International Organization for Standardization、国際標準化機構）といった標準化団体もあり、それぞれさまざまな通信技術の規格を標準化しています。IETFも、それらの標準化団体の一種です。

ただし、IETFにはITUやISOとは大きく違う特徴があります。ITUやISOによる標準化は、de jure standard（デジュールスタンダード、de jureはラテン語で「法律上の」という意味を持つ）と呼ばれます。これは、国家や企業を代表する決められた人々があらかじめ標準を作ってから、多くの人々がそれに従うという、トップダウンな手法です。一方、IETFの標準化は、先に市場でいろいろな技術が登場し、それがde facto standard（デファクトスタンダード、de factoは「事実上の」という意味を持つ）として認められるという手法です[†21]。現在のインターネットで使われている中心的な通信プロトコルであるTCP/IPも、de facto standardのひとつです。

IETFには、誰でも新しい提案を行えることや、誰でも議論に参加できるという特徴もあります。前項で強調したように、インターネットのプロトコルはオープンなので、IETFという場に

[†21] とはいえ、現在のインターネットは世界中に影響を及ぼすので、プロトコルによってはIETFにおけるRFCの策定作業を待ってから利用が本格化するものもあります。そのため、IETFが完全に純粋なde facto standardの場といえるかどうかは、昨今では微妙だといえます。

CC BY NC SA

も誰もが参加できるのです。これも「最初に標準ありき」な他の多くの標準化団体とは異なる点でしょう。

RFCは、なぜ「意見募集」という名前なのか

インターネットの通信技術に関連する調べ物をしたことがある人であれば、RFCという単語を必ず聞いたことがあると思います。**RFC**は、Request For Commentsの略です。これは「意見募集」や「意見を求む」といった意味になります。

インターネットに関連する通信仕様の標準を示す文書の名称が、なぜ「意見募集」なのでしょうか？ その理由は初期のRFCが発行された当時の時代背景から伺えます。RFCには文書番号があり、新しく発行されるたびに文書番号が増やされていくので、基本的にその文書番号が小さいほど昔のものになります。最初のRFCであるRFC 1は1969年4月7日に発行されました。この最初のRFCを発行するときにはさまざまな苦労があったようです。

現在のインターネットを使っていると忘れがちですが、インターネットの前身となる技術の研究は東西冷戦の真っ只中で行われていました。国防上の観点から、当時のDARPA（Defense Advanced Research Projects Agency、アメリカ国防高等研究計画局）によって進められていた研究は「非公開」が前提です。しかし、ARPANET（Advanced Research Projects Agency NETwork、高等研究計画局ネットワーク）を開発していた関係者らは、ネットワーク間を接続する技術は広く公開することが大事だと考えていました。

そこで開発に携わっていた関係者は、周囲を説得するために、「RFCとして発行される文書は非公式であって公式な出版物ではない」としました。“Request For Comments”という名のとおり、「意見を求めるために公開する文章」という建前を掲げたのです。いまでこそ、いろいろな仕様や技術が公開されていることを誰もが当たり前のように享受していますが、このような時代背景から、かつては「公式な技術情報をそのまま公開しているわけではない」とあえて表明することが重要だったことがわかります。

そんな背景を持つRFCですが、実際はRFCとして認められる前に**Internet Draft**という段階があり、この段階で広く意見を求めつつ議論を重ねています。つまり、RFCが発行された時点は「意見募集中」ではなく、すでにさまざまな意見が求められ議論されたあとです[†22]。そのため、現在のRFCは意見を求めるために発行されるものではありません。事実、IETFで毎回行われているNewcomers Orientation（初参加者のための説明会）の中では、「RFCは標準文書という扱いになっているので、意見募集という意味合いはなくなっており、あくまでもRFCというブランド名である」と説明されています[†23]。「かつてのRFCはRequest For Commentsの略だが、いまのRFCはブランドであり公式の出版物である」という説明をIETF自体がしているのです。

世界中の人々が使う現在のインターネットが生まれたのは、通信に関連する仕様がRFCとして公開され、その結果として、相互接続できる機器を多くの人々が実装できたことが大きな要因のひとつです。「RFC」という名称からは、本来であれば非公式だったはずの仕様を公開するための、当時の開発者たちの苦労が垣間見えます。

新しいRFCは現在も次々と作られています。インターネットに関連する仕様は、いまもなお新しいものが登場し、変わり続けているのです。

cc BY NC SA

1.4.3 RFCには何が書かれているか

一口にRFCと言っても、インターネットを運用するうえで有用であるとされる情報にはさまざまなものがあります。つまり、プロトコルの規格を標準化する文章だけがRFCではありません。たとえば、インターネット上での機器の運用方法に関する指針は、標準を決めるという性格の情報ではありませんが、ネットワーク同士がインターネットとして相互接続を安定的に継続し続けるために重要な情報であり、そのような目的のRFCも存在します。インターネットコミュニティが形成されていった過去の経緯など、インターネットコミュニティにとって有益と思える情報を記したRFCもあります。

扱う情報の性質に応じて、RFCは次のようなタイプに分けられます。

- **Standards Track**：プロトコルとしての標準化を目指す文書
- **Informational**：参考情報として公開されている文書
- **Experimental**：実験的な試みとしての手法などが公開されている文書
- **Historic**：歴史的経緯を知る目的で公開されている文書。この分類のRFCを実際に使うことは推奨されていない
- **Best Current Practice**：「その時点における最良の方法」を示す文書。利用方法、活用方法、心得（原則）などプロトコルそのものではないが重要と思われる事柄が示されている

注意が必要なのが、「標準」が必ずしもStandards Trackに限らないという点です。実際に使われているプロトコルなどには、InformationalやExperimentalなRFCのものもあります。たとえば、同じ目的を持ったプロトコルが複数提案されているとき、複数のプロトコルを同時にStandards Trackにするのではなく、それぞれをInformationalやExperimentalとして扱うといった判断がとられることがあります。

IPv6の基本的な仕様はStandards Trackに分類されるものが大半です。ただ、IPv6の運用方法に関連する仕様であったり、IPv6の周辺プロトコルを規定した仕様の中には、Standards Trackではないものもあります。

もう一点注意が必要なのが、RFCに書かれていたとしても、それが現実世界で採用されているとは限らないという点です。たとえば、Standards TrackのRFCに書かれていることが、すべてそのままの形で実装されるとは限らないのです。誰も実装すらしていない場合もあります。本書ではRFCに書かれている内容について数多く解説していますが、本書執筆時点において実装や運用が進んでいないものもあるという点には注意してください。

Joke RFC

RFCの中には、インターネットの運用には何の役にも立たないものもあります。特に、4月1日に発行される**Joke RFC**は、**エイプリルフールRFC**とも呼ばれています。

---

†22 IETFの議論を経ずにRFCが発行される場合もあります。
†23 https://trac.ietf.org/trac/edu/attachment/wiki/IETF95/95-newcomers.ppt

CC BY NC SA

Joke RFCは、Informational RFCに分類されます。有名なJoke RFCとしては、1990年4月1日に発行されたRFC 1149[†24]（鳥類キャリアによるIP、一般的には「IP over 伝書鳩」として知られている）などが挙げられます。Joke RFCには、技術的な背景を持ちつつ、どこかしら愛せるアホらしさが求められます。

Joke RFCに実用的な利点があるとすれば、世界中の技術者に一瞬だけ心のオアシスを提供することぐらいだろうと思いますが、冗談までもが「標準」と並列して扱われる標準化団体はIETFぐらいのものでしょう。

1.4.4　IETF以外からのRFC

ここまでの説明では、RFCを議論するのはIETFであると紹介してきましたが、IETF以外で作られた文書がRFCとして発行される場合もあります。RFCが発行されるまでの流れ（**ストリーム**）として、RFC 4844[†25]では以下の4種類が定義されています。

- IETFストリーム
- IABストリーム
- IRTFストリーム
- 個人による投稿からのストリーム

IABストリームに関してはRFC 4845[†26]、IRTFストリームに関してはRFC 5742[†27]、個人による投稿からのストリームに関してはRFC 4846[†28]でそれぞれ具体的に紹介されています。

1.4.5　RFCの廃止や更新

一度発行されたRFCは、内容が変更されず、そのままの状態で公開され続けますが、時代とともにRFCに書かれた内容が古くなることもあります。そのため、RFCを廃止したり更新したりする方法も決められています。具体的には、一度発行されたRFCは内容を変更できないので、別のRFCによって内容を更新されたり、廃止されたりすることになっています[†29]。

IPv6の基本仕様が記されたRFCも、これまで何度か上書き廃止されています。RFCを調べるときは、RFCに記述してあることが最新であるかどうかに気をつけることも、非常に大事です。

■ RFC 2460を廃止したRFC 8200

本書執筆時点におけるIPv6基本仕様のRFCは、2017年7月に発行されたRFC 8200です。RFC 8200は、それまでのIPv6基本仕様を示していたRFC 2460[†30]を廃止して上書きするRFC

---

†24　RFC 1149：D. Waitzman, “Standard for the transmission of IP datagrams on avian carriers”, 1990年4月
†25　RFC 4844：L. Daigle, Internet Architecture Board, “The RFC Series and RFC Editor”, 2007年7月
†26　RFC 4845：L. Daigle, Internet Architecture Board, “Process for Publication of IAB RFCs”, 2007年7月
†27　RFC 5742：H. Alvestrand, R. Housley, “IESG Procedures for Handling of Independent and IRTF Stream Submissions”, 2009年12月
†28　RFC 4846：J. Klensin, D. Thaler, “Independent Submissions to the RFC Editor”, 2007年7月
†29　古いRFCを廃止する新しいRFCが古いRFCと同じタイトルの場合もあれば、タイトルが微調整されるなど、異なるものになることもあります。
†30　RFC 2460：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 1998年12月

です。RFC 2460が発行されたのは1998年なので、IPv6の基本仕様を規定したRFCが廃止されるのは約19年間ぶり、2回めの廃止です[†31]。

RFC 2460が廃止された背景として、IETFにおけるRFCの「標準」の扱いが変化したことが挙げられます。

RFCによる標準化のプロセスは、RFC 2026[†32]として定義されています。RFC 2026には、Proposed Standard（標準化に向けた提唱）、Draft Standard（標準化への草稿）、Internet Standard（インターネット標準）という3つのレベルが定められていますが、IPv6基本仕様を定義しているRFC 2460は、このうちのDraft Standardでした。RFC 2460は、RFC的な位置づけとしては、「Internet Standard」（インターネット標準、単にStandardと表現されることもあります）ではなく、「Draft Standard」だったのです。

しかし、2011年に発行されたRFC 6410[†33]がRFC 2026を更新することによって、RFCによる標準化のプロセスが変わりました。具体的には、RFC 2026で示されている3つのRFCのレベルが減り、Proposed StandardとInternet Standardの2段階へと減りました。それまで存在していたDraft Standardが冗長だったので、提案（Proposed Standard）の次の段階が標準（Internet Standard）へと変更されたのです。

RFC 6410では、Draft Standardというレベルを廃止すると同時に、既存のDraft StandardのRFCの扱いについても定義されています。それによると、それまでDraft StandardだったRFCは、以下のような方針でProposed Standardに降格かInternet Standardへ昇格できるとされています。

- すでに使われている仕様との不整合を起こす不具合がDraft Standardの仕様に含まれていないのであれば、Internet Standardへと再分類できる
- RFC 6410が発行されてから2年後以降は、IESGによってDraft StandardのRFCをProposed Standardへと再分類できる

IPv6の仕様については、「草稿」であるDraft Standardから「標準」であるInternet Standardへと昇格させるため、2015年からIETFの6manワーキンググループにて議論が開始されました。その議論を経て作られたのがRFC 8200です。

RFC 8200の基本的な内容はRFC 2460と大きくは変わりません。RFC 2460からの変更点がRFC 8200のAppendix Bに記載されていますが、主な違いとして挙げられているのは、RFC 2460に対して別のRFCが更新している内容の反映、曖昧であった部分の明確な表現への修正、IPv6拡張ヘッダに関連する仕様変更などです。細かいところでは、“IP Next Generation”という表現が削除されました。かつてIPv6は「次世代インターネットプロトコル」とされていましたが、もう「次世代」ではないということで、その表現が削除されたのです。

仕様としての変更点という視点で見ると、RFC 8200は「それまでにたまった各種の仕様変

---

†31 本書の執筆を始めた段階ではRFC 2460をベースにしていましたが、途中からRFC 8200をベースにして書き換えました。

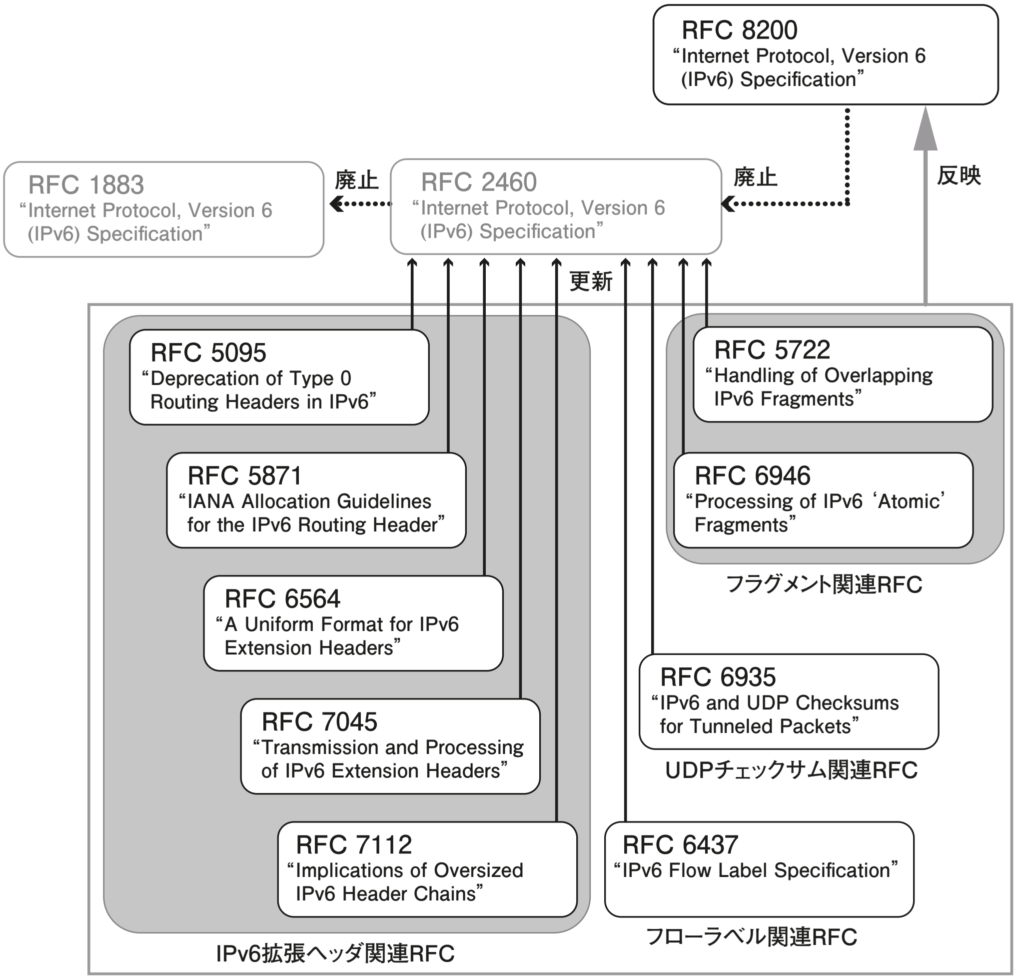
†32 RFC 2026：S. Bradner, “The Internet Standards Process – Revision 3”, 1996年10月

†33 RFC 6410：R. Housley, D. Crocker, E. Burger, “Reducing the Standards Track to Two Maturity Levels”, 2011年10月

更を反映したもの」という側面が強いといえます。RFC 8200が発行される前のRFC 2460に対しては、図1.23のように、さまざまな更新のためのRFCが発行されています。

なお、RFC 2460も別のRFCを廃止するRFCです。1995年にRFC 1883[†34]が最初のIPv6基本仕様をまとめたRFCとして発行され、それが1998年にRFC 2460の発行によりリニューアルされています。

RFC 8200では、RFC 2460に対して行われていたこれらの変更が反映されています。RFC 8200が発行されるまで、IPv6を学ぶ人は、RFC 2460を読みつつ、RFC 2460を更新するさまざまなRFCを個別に読む必要がありました。「RFC 2460には、こう書いてあるけど、RFC ◯◯がRFC 2460を更新しているから、その部分に関してはRFC 2460に書いてあることをそのまま覚えてはダメだ」という状況が続いていたのです。RFC 8200によってそのような状況は改善されています。



▶図1.23　RFC 8200が発行される前のRFC 2460

†34　RFC 1883：S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", 1995年12月

CC BY NC SA

IETFにおけるIPv4の位置づけの変化

2016年11月7日に、IETFにおける標準化においてIPv4を前提とせず、IPv6を前提とするとした声明がIABから発表されました。

- IAB Statement on IPv6：
  https://www.iab.org/2016/11/07/iab-statement-on-ipv6/

この声明では、当面はIPv4とIPv6のデュアルスタックや移行技術（共存技術）などの「IPv4との後方互換を実現する環境」が必要であることは認めつつ、標準化団体にとって大事なのは「IPv4からIPv6への移行の障害となったり減速させたりする標準化を行わないこと」であると述べています。また、IETFで議論される新しいプロトコルや既存のプロトコルの拡張においてIPv4に対する互換性を要求し続けることをせず、IPv6に最適化されたものにするともあります。

このように、IETFではIPv6を前提とした標準化が進んでおり、IPv4に対する位置づけが昔とは変わっています。

1.4.6　番号と名前を管理するIANA

世界中でインターネットを破綻なく利用するためには、RFCとしての仕様だけではなく、通信に必要となる特定の番号や名前などを複数の異なる組織が重複して利用してしまわないための仕組みも必要です。インターネットで利用する番号や名前などの「資源」を統一的に管理する機能は、**IANA**（Internet Assigned Numbers Authority）と呼ばれています。日本語に直訳すると、「インターネットにおいて割り当てられる番号に関する権威」という意味になります。

もともとのIANAは、ARPANET（インターネットの前身であった研究用ネットワーク）に携わっていたジョン・ポステル氏らがインターネットに必要な資源を管理していた副産物でした。インターネットが急速に拡大するとともに、番号などのインターネット資源をどのように管理すべきかが議論された結果、ICANN（Internet Corporation for Assigned Names and Numbers）という非営利団体が創設され、ICANNがIANA機能を運用する形になりました。2016年8月以降は、ICANNが米国カリフォルニア州で設立した非営利公益法人であるPublic Technical IdentifiersがIANA機能を運用しています。

IANAが管理している主なインターネットの資源は、ドメイン名に関する情報、番号資源、プロトコルパラメータの3つです。

- ドメイン名に関する情報

  インターネットでは、名前とIPアドレスの対応をDNS（Domain Name System）によって解決しています。DNSによる名前解決の仕組みで最も重要な役割を担うルートサーバに登録される情報は、IANAによって管理されています。DNSについては第16章で詳しく説明します。

  ルートサーバの情報のほか、特殊な用途のトップレベルドメインや、`example.com`などの例示用ドメインをはじめとする特殊なドメイン名、DNSSECのルート鍵署名鍵（Root Key Signing Key）などもIANAが管理しています。

cc BY NC SA

- **番号資源**

  インターネット全体で使うIPアドレスに重複があってはいけません。誰がどこでどのようなIPアドレスを使うかは、IANAを頂点として階層的に一元管理されています。本章の主題であるIPv6アドレス空間もIANAで管理しています。IANAが管理するIPv6アドレス空間については3.2節で説明します。

  IPアドレスのほか、ルーティングプロトコルのBGPで利用するAS（自律システム、Autonomous System）を一意に表すための番号もIANAが管理しています。

- **プロトコルパラメータ**

  IPアドレスやAS番号だけでなく、さまざまなプロトコルを表す番号や、各プロトコル内で利用される番号や文字列の多くもIANAで管理されています。IANAが管理するプロトコルパラメータの一覧は`https://www.iana.org/protocols`で参照できます。

CC BY NC SA

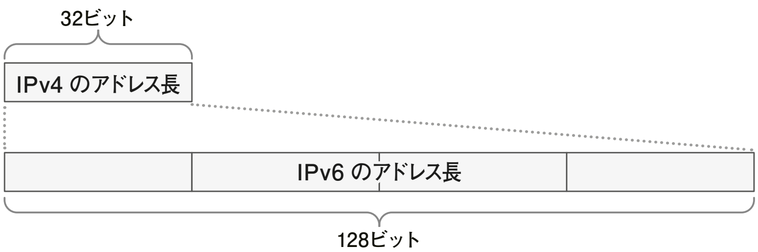
第2章

IPv6概論

本章ではIPv6の基本的な仕組みについて概要を紹介します。IPv6を構成するさまざまな要素について大まかな枠組みを整理し、それぞれの詳細については第II部で改めて解説します。

2.1 IPv6アドレス

IPv6はIPv4アドレスの在庫枯渇問題に対する解決策として作られました。そのような背景もあり、IPv6とIPv4の違いとして最もわかりやすいのは、インターネットで利用できるアドレスの個数を左右するアドレス長だといえます。IPv4のアドレス長は32ビット、IPv6のアドレス長は128ビットです。



▶ 図2.1　IPv4アドレスとIPv6アドレスのビット長

本節では、128ビット長のIPv6アドレスによってもたらされるアドレス空間について説明します。この128ビットのIPv6アドレスを人間が認識できる形でどのように表記するかについては次節で紹介します。

2.1.1 IPv6アドレス空間

IPv4アドレスとIPv6アドレスを比べると、32ビットと128ビットなので、長さとしては4倍です。「何だ、たったの4倍か」と思うかもしれませんが、あくまでも長さが4倍になるのであり、アドレス空間の大きさ、つまり利用できるIPアドレスの総数は4倍どころではありません。32ビットのアドレス空間を4倍にするだけならば、アドレスの長さは128ビットも必要なく、34ビットで十分です。

アドレス空間は、アドレスの長さを1ビット増やすたびに、2倍ずつ膨らみます。32ビットと128ビットでは、アドレスの長さは96ビット増えたことになります。したがって、アドレス空間が32ビットから128ビットに増えるということは、IPv6アドレスで表現可能なアドレスの総数がIPv4アドレスのときに比べて2の96乗倍になるということなのです。

2.1.2 実際に使えるIPv6アドレス空間

2の96乗倍は、天文学的な数字です。そのため、よく「IPv6には事実上無限ともいえるアドレス空間がある」と表現されることがあります。実際、128ビットというアドレス長の数字だけで考えると、確かにIPv6には128ビット分のアドレス空間があり、それだけの数のIPアドレスを表現可能です。

とはいえ、その広大なアドレス空間から好き勝手にアドレスを使えるわけではありません。IPv6アドレスはIPv4アドレスとは比較にならないぐらいジャブジャブと運用されていることもあり、実際に使えるIPv6アドレスの数は限られています。

まず、2.3節で後述するように、IPv6のアドレスのうち上位64ビットはネットワークアドレスとして利用されます。したがって、各ネットワーク内で表現可能なIPアドレスの数は、残りの64ビットで表現可能な個数になります。具体的には、各ネットワーク内で「IPv4インターネットで表現可能なアドレスの総数」の約43億倍が利用可能です。

これは、家庭内などの各ネットワークで利用可能なIPv6アドレスだけで、いまのIPv4インターネット全体で利用可能なアドレスの総数の約43億倍もあることを意味します。筆者には、1つの家庭内でそれだけのIPv6アドレスが必要になるユースケースがすぐには思いつきません（外部からの単純なポートスキャンは多少やりにくくなりそうですね）。実際の運用上は使われないままになるIPv6アドレスがかなり多くなると考えられます。

各事業者に割り振りされるIPv6アドレスブロックのサイズも、IPv4とは比べものにならないほど巨大です。たとえば、1つの事業者に/20のIPv6アドレスブロックが割り振られることもあります。

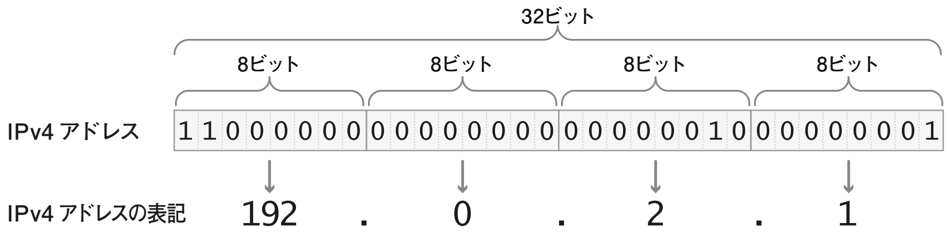
こう考えると、IPv6のアドレス空間は無限にあるように見えて、実際の運用で利用されるアドレスはかなり少ないと推測できます。IPv6がIPv4のように枯渇するという状況は、いまのところ想像しにくいですが、かといって事実上無限のアドレスがあるかというと、「それは言いすぎではないか？」というのが現在の筆者の考えです。

2.2 IPv6アドレスのテキスト表記

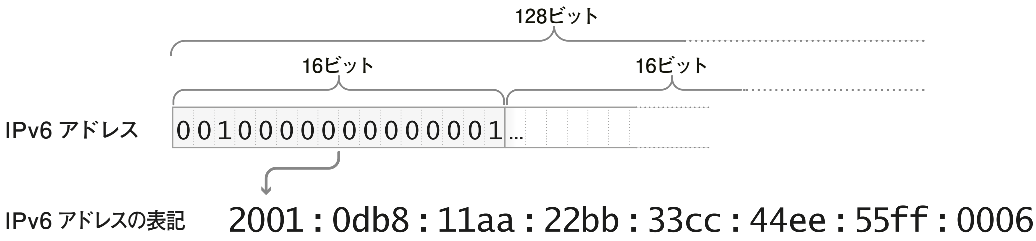
IPv6アドレスをコンピュータの中で扱うときには、単に128ビットの情報として処理すればよいのですが、その128ビットを人間が理解しやすい文字列で表記するには少し工夫が必要です。IPv4の場合は、「`192.0.2.100`」のように、8ビットごとに「`.`」（ドット）で区切るドット付き十進表記（dotted decimal notation）が一般的でした。

IPv6の場合は、IPv4のドット付き十進表記と同じ方法では長くなりすぎてしまうので、「`2001:db8:11aa:22bb:33cc:44dd:55ee:66ff`」のように、16ビットごとに「`:`」（コロン）で区切って4ビットごとに十六進表記することになっています。

CC BY NC SA



▶ 図 2.2　IPv4 アドレスのテキスト表現（ドット付き十進表記）の例



▶ 図 2.3　IPv6 アドレスのテキスト表現の例

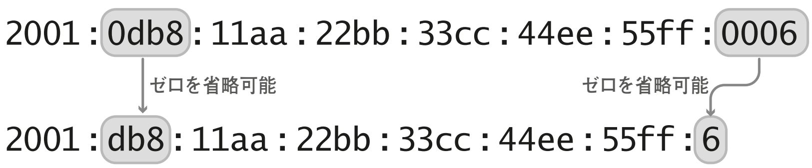
2.2.1　IPv6 アドレスの省略表記

図 2.3 の表記は、2006 年に発行された RFC 4291[†1]にて示されているものです。ここで注意が必要なのは、RFC 4291 にて定義されている同じ IPv6 アドレスに対し、複数の表記方法が存在する場合があることです。それら複数の表記方法を統一して単純化する方法として、2010 年に RFC 5952[†2]が発行されています。

■ RFC 4291 の表記ルール

まず、RFC 4291 で定義されている IPv6 アドレスのテキスト表記ルールを説明します。

- IPv6 アドレスは、16 ビットごとに区切った十六進表記で表されるが、それぞれの 16 ビットのフィールドの先頭で連続する 0 は省略できる（図 2.4）

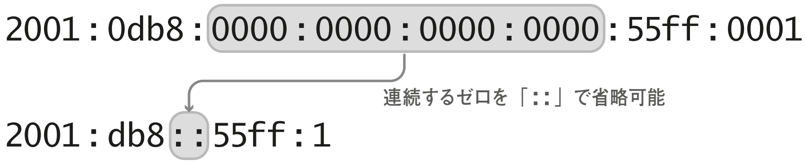


▶ 図 2.4　IPv6 アドレスのテキスト表現 1

- 複数の 16 ビットフィールドにわたって 0 が続く場合、「::」という表記で省略してもよい（図 2.5）

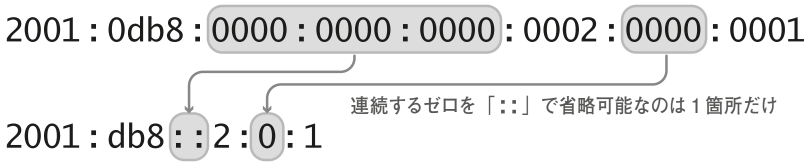
†1　RFC 4291：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 2006 年 2 月

†2　RFC 5952：S. Kawamura, M. Kawashima, “A Recommendation for IPv6 Address Text Representation”, 2010 年 8 月



▶ 図 2.5　IPv6 アドレスのテキスト表現 2

- ただし、「::」が適用可能な 16 ビットフィールドが複数存在するときは、「::」を利用して省略できるのは 1 箇所のみ（図 2.6）



▶ 図 2.6　IPv6 アドレスのテキスト表現 3

さらに、RFC 4291 では、IPv4 アドレスを IPv6 アドレス内に含む特殊な IPv6 アドレスの表記も示されています。RFC 4291 では、IPv6 アドレスのうちの上位の 16 ビットフィールド 6 個を 16 進数で示し、下位 32 ビットを IPv4 アドレスの表記によって 10 進数で示しています。以下は、RFC 4291 で実際に示されている例です。

- `0:0:0:0:0:0:13.1.68.3`
- `0:0:0:0:0:FFFF:129.144.52.38`

このアドレス表記も圧縮が可能です。

- `::13.1.68.3`
- `::FFFF:129.144.52.38`

■ RFC 5952 のルール

RFC 4291 の IPv6 アドレス表記ルールでは、同じ IPv6 アドレスに対して複数の表記方法が可能となる場合があります。たとえば、RFC 4291 のルールで同じ IPv6 アドレスに複数の表記が可能になる例として、RFC 5952 では以下のような IPv6 アドレスが例示されています。一見するとそれぞれ異なるものに見えますが、すべて同じ IPv6 アドレスに対する RFC 4291 のルールに従った表記です。

- 2001:db8:0:0:1:0:0:1
- 2001:0db8:0:0:1:0:0:1
- 2001:db8::1:0:0:1
- 2001:db8::0:1:0:0:1
- 2001:0db8::1:0:0:1
- 2001:db8:0:0:1::1

- 2001:db8:0000:0:1::1
- 2001:DB8:0:0:1::1

このような曖昧さを排除し、IPv6アドレスの統一されたテキスト表現を規定しているのが、2010年8月に発行されたRFC 5952です。RFC 5952では以下のルールが規定されています。

- 「::」による短縮時には可能な限り短くする
  「16ビットの0」が連続していて「::」による短縮が可能な場合には、最も短縮できる表現を利用しなくてはなりません。たとえば、2001:db8:0:0:0:0:2:1というIPv6アドレスは2001:db8::2:1と表現すべきであり、2001:db8::0:2:1などと表現してはいけません。
- 「16ビットの0」が1つだけの場合は「::」を利用してはならない
  「::」による短縮は、「16ビットの0」が連続して出現する場合にしか使えません。たとえば、2001:db8:0:1:1:1:1:1を2001:db8::1:1:1:1:1とは表記できません。
- 最も短縮できる箇所で「::」を利用する
  たとえば、2001:0:0:1:0:0:0:1というIPv6アドレスには、「16ビットの0」が2つ連続する箇所と、3つ連続する箇所があります。最も短縮できるのは3つ連続した「16ビット0」なので、2001:0:0:1::1と表記します。
- 同じ長さの「16ビットの0」が複数ある場合は、最初の箇所で「::」を利用する
  たとえば、2001:db8:0:0:1:0:0:1というIPv6アドレスには、「16ビット0」が2つ連続する箇所が2つあります。最初に登場する16ビットの0を「::」で短縮して、2001:db8::1:0:0:1と表記します。
- アルファベットは大文字ではなく小文字を利用する
  十進表記における10〜15は、十六進表記ではアルファベット1文字で表しますが、このとき大文字ではなく小文字を利用します。つまり、abcdefを利用し、ABCDEFを利用しないようにしてください。
- 特殊なIPv6アドレスでの利用
  3.9節で説明するIPv4-Mapped IPv6アドレスでは、::ffff:192.0.2.1のようなテキスト表現を利用します。つまり、先頭にある0が続く部分に関しては「::」で省略し、「0:0:0:0:0:ffff:192.0.2.1」のようには表現しないことが求められます。
- ポート番号との表記
  IPv4では、アドレスのテキスト表現に続けて「:」区切りでポート番号を併記する方法があります。IPv6でもアドレスにポート番号を併記することができて、RFC 5952には下記のような例が紹介されています。
    - [2001:db8::1]:80
    - 2001:db8::1:80（この方法は推奨されません）
    - 2001:db8::1.80

- `2001:db8::1 port 80`
- `2001:db8::1p80`
- `2001:db8::1#80`

上記のうち、RFC 5952で推奨されているのは、RFC 3986[†3]で示されているIPv6アドレス部分を「[]」（角括弧）で囲んだ一番上の例です。逆に、IPv6アドレスとポート番号を「:」で区切る上から2つめの表現は非推奨とされています。

RFC 3986で示されている[]でIPv6アドレス部分を囲んだ表現は、Webブラウザをはじめ多くのアプリケーションで採用されています。RFC 5952には、「IPv6アドレスを含むURIを扱う場合にはRFC 3986に準拠する必要がある」と書かれています。

■ RFC 5952のルールは人間向け

RFC 5952のルールは、人間が混乱しないための表記方法を提案するものであり、データベース内などで利用することを前提としたものではありません。テキストとしてIPv6アドレス情報を表現するときに読みやすい方法にしましょうという話であって、文字列をパースするときの条件を示したものではないのです。

そのため、文字列で表現されたIPv6アドレスをパースするアプリケーションを作る場合には、RFC 4291に従った表記を正しく処理できるようにする必要があります。

IPv6アドレスのテキスト表記に名前がついてない

IPv4アドレスを示すために最もよく使われるのが、十進表記による数値を「.（ドット）」で区切ったテキスト表記ですが、その表記方法には「ドット付き十進表記」（“dotted decimal notation”）という名前があります。一方、IPv6アドレスのテキスト表記には、明確な名前が存在していないようです。

少なくとも、IPv6の基本仕様を規定したRFC 8200、IPv6アドレス体系の基本仕様であるRFC 4291、推奨されるIPv6アドレスのテキスト表記に関するRFC 5952には、IPv6アドレス表記そのものの名前を示した表現は記載されていません。十六進表記であったり、コロン区切りであったりという説明のほかに、単に“IPv6 Address Text Representation”とあるぐらいなのです。

多少関連する話として、以前、コロンで区切られる16ビット部分のために“**hextet**”という表現がIETFで提案されたことがあります[†4]。この“hextet”という表現を使って、IPv6アドレスのテキスト表記を“colon separated hextet”（コロン区切りhextet）と表現することも考えられます。実際、Webで検索してみると、いくつかそのような表現も発見できます。しかし、“hextet”という表現を公式にIETFで扱うという提案は採用されていないので、IPv6の各種RFCには“hextet”という単語は登場していません。これが公式な表現として広まることはなさそうです。

非公式な呼び方として「コロン区切り十六進表記」といった表現を用いることも考えられるでしょう。しかし、本書執筆時点のRFCのように単に「IPv6アドレスのテキスト

---

†3　RFC 3986：T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, 2005年1月

CC BY NC SA

表記」という表現を使えば十分に伝わるので、あえて特別な呼び方を使うことはむしろ混乱を招くとも考えられます。

そう考えてみると、そもそもなぜIPv4アドレスのテキスト表記には “dotted decimal notation” という名前がつけられたのでしょう？ こちらのほうが謎に思えてきます。

2.3 IPv6アドレス体系

128ビットのIPv6アドレスには、さまざまな種類や意味に応じて構造が定義されている場合があります。IPv6アドレスの種類とフォーマットは、IPv6アドレス体系（IPv6 Addressing Architecture）として、RFC 4291にまとめられています。

ここではIPv6アドレス体系の概要を紹介します。詳細について第3章で改めて解説します。

IPv6 のアドレス体系を規定した RFC は何度も更新されている

IPv6のアドレス体系を規定したRFCは、これまで何度も改訂されています。細かいところでいろいろな違いが発生している部分なので、2000年代前半に勉強した知識のままである場合には注意が必要です。下記に、IPv6アドレス体系に関するRFCの変遷をまとめます。

- RFC 1884[†5]: IP Version 6 Addressing Architecture（1995年）
- RFC 2373[†6]: IP Version 6 Addressing Architecture（1998年）
- RFC 3513[†7]: Internet Protocol Version 6 (IPv6) Addressing Architecture（2003年）
- RFC 4291: IP Version 6 Addressing Architecture（2006年）

いまのところ、2006年に制定されたRFC 4291が廃止されるような動きはありません[†8]。RFC 4291は、それ以前の3つのRFCよりは安定しています。

2.3.1 IPv6アドレスの種類

RFC 4291では、IPv6アドレスはユニキャスト、エニーキャスト、マルチキャストの3種類に分類されるとあります。そのうえで、IPv6アドレスの基本的な種類を下記のように定義しています（個々のアドレスの詳細については3.1節で改めて説明します）。

- 未定義アドレス（`::/128`）

---

†4 Naming IPv6 address parts : https://tools.ietf.org/html/draft-denog-v6ops-addresspartnaming-04

†5 RFC 1884：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 1995年12月

†6 RFC 2373：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 1998年7月

†7 RFC 3513：R. Hinden, S. Deering, “Internet Protocol Version 6 (IPv6) Addressing Architecture”, 2003年4月

†8 IETFの6man WGでRFC 4291を上書き廃止するためのInternet Draftが議論されていましたが、2018年1月にExpireになりました。https://www.ietf.org/archive/id/draft-ietf-6man-rfc4291bis-09.txt

- ループバックアドレス（::1/128）
- マルチキャストアドレス（ff00::/8）
- リンクローカルユニキャストアドレス（fe80::/10）
- グローバルユニキャストアドレス（上記以外のすべてのアドレス）

NOTE

エニーキャストではユニキャストアドレスを使うので、IPv6アドレスを見ただけではユニキャストかエニーキャストかは判別できません。なお、ユニキャストについては第3章、エニーキャストについては第12章、マルチキャストについては第11章でそれぞれ改めて解説します。

2.3.2 IPv6アドレスの利用モデル

IPv6アドレス体系を規定するRFC 4291では、「IPv6アドレスがどのように利用されるのか」について“2.1 Addressing Model”で説明しています。そこでは、まずIPv6アドレスは「ノードではなくネットワークインターフェースに対して設定される」としています。なお、ネットワークインターフェースは必ずノードに所属しているので、結果としてネットワークインターフェースによってノードも識別可能です。

さらにRFC 4291では、「IPv6ではすべてのネットワークインターフェースに最低でも1つのリンクローカルユニキャストアドレスを設定する必要がある」としています。そして、「1つのネットワークインターフェースに対して複数のIPv6アドレスを追加できる」ことが明記されています。

サブネット（プレフィックス長）については、単一のリンクに関連づけられるという点で、IPv6でもIPv4のときと同様です。ただし、IPv6では単一のリンクに対して複数のサブネットを関連づけることができるという点が異なっています。

1つのネットワークインターフェースに複数のIPアドレスを設定できることと、同一リンク上に複数のサブネットが同時に存在できることは、IPv6の大きな特徴です。本書の以降の説明でも、これらの性質がIPv6にもたらしている機能について繰り返し言及していきます。

■ リンクローカルユニキャストアドレス

IPv6では、ネットワークインターフェースに対し、同一リンク内でのみ有効となるIPv6アドレスが設定されます。これをリンクローカルユニキャストアドレスと呼びます。fe80から始まるfe80::/10がリンクローカルユニキャストアドレスです。

IPv6を利用するすべてのネットワークインターフェースには、リンクローカルユニキャストアドレスを設定する必要があることが定められています。実際、IPv6が有効な状態でネットワークインターフェースを初期化すると、リンクローカルユニキャストアドレスが自動的に設定されることがほとんどです。

リンク内に限定された通信プロトコル、すなわちルータを越えることが想定されていない通信では、基本的にリンクローカルユニキャストアドレスを使います。IPv6の基本的な機能である近隣探索プロトコル（第6章）やIPv6アドレスの自動設定（第7章）、ルーティングプロトコルであるOSPFv3なども、ネットワークインターフェースにリンクローカルユニキャストア

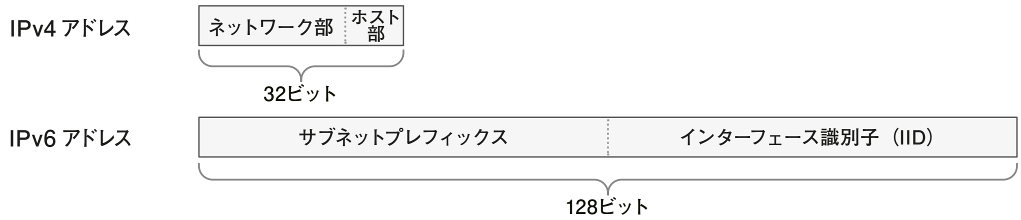
CC BY NC SA

ドレスが設定されていることを前提とした設計になっています。

リンクローカルユニキャストアドレスについては3.5節で詳しく解説します。

2.4 IPv6におけるユニキャストアドレスの構成要素

1.1.4項で説明したように、IPアドレスは「ネットワークを示す部分」と「インターフェースを示す部分」とに分けられます。この区分はIPv4とIPv6とで異なります（図2.7）。



▶ 図2.7 IPv4とIPv6でのIPアドレス構成要素の違い

IPv4では、IPアドレスはネットワーク部とホスト部によって構成されます。一方、IPv6では、IPアドレスはサブネットプレフィックスとインターフェース識別子（IID、Interface Identifier）によって構成されます。この違いは、一見すると名称の問題だけのようにも思えますが、IPv4とIPv6の大きな違いを示しています。

1989年に発行されたRFC 1122[†9]では、IPv4アドレスの構成要素を“Network-number”と“Host-number”としています。つまり、IPv4アドレスは、機器である「ホスト」そのものを示しています。ルータ以外の末端ノードは基本的に1つのネットワークインターフェースだけを持ち、そこで必要なIPv4アドレスも1つだけという前提が垣間見えます。

一方、IPv6では、1つのネットワークインターフェースに対して複数のIPv6アドレスが付く場合が最初から想定されています。IPv6アドレスにおける「インターフェース識別子」という名称は、文字どおり、ホスト[†10]ではなくインターフェースを識別するという表現です。インターフェース識別子という名称からは、IPv6アドレスはあくまでもネットワークインターフェースを識別するものであり、必ずしも機器を識別するものではないという前提が垣間見えます。実際、IPv6では、同じ1つのネットワークインターフェースに対して「サブネットプレフィックスは異なるがインターフェース識別子は同じ」というIPv6アドレスを複数設定できます。

NOTE

一部の例外はあるものの、基本的にはユニキャストおよびエニーキャストのIPv6アドレスの上位64ビットはサブネットプレフィックス、下位64ビットはインターフェース識別子です。

†9 RFC 1122：R. Braden, “Requirements for Internet Hosts - Communication Layers”, 1989年10月

†10 IPv6の基本仕様を示したRFC 8200での定義では、「ホスト」はルータではないノード、と定義されています。1.1.3項も参照。

CC BY NC SA

2.5 IPv6基本プロトコル

プロトコルとしてのIPv6を規定しているのは、“Internet Protocol, Version 6 (IPv6) Specification”（「インターネットプロトコル, バージョン6（IPv6）仕様」）と題されたRFC 8200[†11]です。RFC 8200の冒頭の章（“1. Abstract”）には、「このドキュメントはインターネットプロトコルのバージョン6を規定している」と極めてシンプルに記されています。

RFC 8200に記載されているのは、基本的にはIPv6ヘッダ（およびIPv6拡張ヘッダ）のフォーマットに関する内容です。それ以外にも、パケットのサイズ、上位レイヤーにおけるIPv6対応、セキュリティ、フローラベル、トラフィッククラスなどに関する記述がありますが、特にセキュリティ、フローラベル、トラフィッククラスに関しては外部参照を含めた紹介に留まっており、IPv6ヘッダがIPv6基本プロトコルそのものだといえるでしょう。

IPv6ヘッダの構造については、IPv4ヘッダと比較した特徴を含め、第4章で詳しく解説します。

2.6 ICMPv6の役割

ICMPv6は、IPv6のためのInternet Control Message Protocolです。その名のとおり、インターネットに関連するコントロールを行うためのメッセージを扱うプロトコルです。

ICMPv6の役割は大きく分けて2つあります。パケットを処理する際に発生したエラーを伝えることと、インターネット層に関する機能を実現することです。到達性を確認するために使われることが多い`ping`コマンド（`ping6`）でもICMPv6を利用しています。

ICMPv6はRFC 4443[†12]で規定されており、その実装はIPv6ノードにとって必須とされています。近隣探索プロトコルや、マルチキャストの制御を行うMLD（Multicast Listener Discovery）など、IPv6を支えるさまざまな仕組みの土台にもなっているプロトコルです（ICMPv6を土台とする他のプロトコルについてはRFC 4443とは別のRFCとしてそれぞれ規定されています）。

本書では、ICMPv6そのものについては第5章、近隣探索プロトコルについては第6章、MLDについては11.5節でそれぞれ解説します。

2.7 近隣探索プロトコル

RFC 4861で規定されている近隣探索プロトコル（Neighbor Discovery Protocol）は、IPv6の根幹を支える重要な仕組みのひとつです。プロトコルの名称にある「近隣」とは同一リンクのことを指しており、それを規定しているのがRFC 4861だといえます。RFC 4861では、ホストが近隣に存在する他のノードに関連する情報をどのように扱うか、同一リンク内でどのようにパケットを送信すべきかなどについて、“Conceptual Model of a Host”（ホストの概念モデル）という枠組みによって説明しています。

近隣探索プロトコルが実現する機能には次のようなものがあります。

---

†11　RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017年7月

†12　RFC 4443：A. Conta, S. Deering, M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, 2006年3月

cc BY NC SA

- リンク上のルータを探す機能（Router Discovery）
- ルータを経由せずに到達できるIPv6アドレスの範囲を知る機能（Prefix Discovery）
- リンクのMTUなどの情報を知る機能（Parameter Discovery）
- インターフェースに対してステートレスにアドレスを振る機能（Address Autoconfiguration）
- IPv6アドレスからリンク層のアドレスを解決する機能（Address Resolution）
- 宛先アドレスをもとに、次にパケットを送出すべきIPv6アドレスを知る機能（Next-hop Determination）
- 近隣ノードに到達できなくなったことを知る近隣不到達性検知（Neighbor Unreachability Detection）
- 利用するアドレスが他のノードで使われていないかを確認する機能（Duplicate Address Detection）
- ルータからホストへ、より適切な送出先を伝える方法（Redirect）

近隣探索プロトコルでは、以下の5種類のICMPv6メッセージを利用することで、同一リンクにおけるIPv6の通信に必要なさまざまな機能を実現しています。

- Router Solicitationメッセージ
- Router Advertisementメッセージ
- Neighbor Solicitationメッセージ
- Neighbor Advertisementメッセージ
- Redirectメッセージ

2.8 IPv6アドレスの自動設定

IPv6の大きな特徴のひとつとして、IPv6アドレスを生成してそれを自動設定する機能があります。IPv6アドレスの自動設定には、大きく分けて、近隣探索プロトコルによるSLAACとステートレスDHCPv6を併用する方法、SLAACとRDNSSオプションを利用する方法、ステートフルDHCPv6を利用する方法の3つがあります。

SLAAC（StateLess Address AutoConfiguration、「すらっく」と発音します）は、近隣探索プロトコルで定義されているRouter Advertisementメッセージ（RA）に含まれるプレフィックス情報を利用して、ホストのIPv6アドレスを自動設定する方法です。ホストは、RAから得られたプレフィックス情報をサブネットプレフィックスとし、それに自ら生成したインターフェース識別子を組み合わせてIPv6アドレスを自動生成します。インターフェース識別子を自動生成する方法を含め、SLAACの詳細については第7章で解説します。

SLAACが開発された当初は、ホストが名前解決を行うためのキャッシュDNSサーバ（もしくはRecursive DNSサーバ）のIPv6アドレスをRouter Advertisementメッセージに含むことができませんでした。そのため、キャッシュDNSサーバなどの情報を伝えるための仕組みとしてステートレスDHCPv6が開発されました。SLAACとステートレスDHCPv6を同時に使うことで、「IPv6アドレスはホストで自動生成しつつ、キャッシュDNSサーバなどの情報をステー

CC BY NC SA

トレス DHCPv6 で得る」という環境が実現できます。

SLAAC による IPv6 アドレス自動設定は、ステートレス DHCPv6 を併用しない方法でも実現できます。IPv6 が開発された当初は存在しなかった手法ですが、後に Router Advertisement メッセージにキャッシュ DNS サーバの情報を追加できる RDNSS オプションが登場したことにより、SLAAC だけでホストの IPv6 アドレスを自動設定するための環境をすべて用意できるようになりました。

SLAAC と併用されるステートレス DHCPv6 は、サーバがクライアントの質問に対して返答するだけのステートレスでシンプルな仕組みです。これに対し、DHCPv6 を使って、IPv4 の DHCP と同様にサーバがクライアントの IPv6 アドレスを管理することもできます。このように DHCPv6 でホストに IPv6 アドレスを割り当てる方法はステートフル DHCPv6 と呼ばれています。

NOTE

IPv6 には、さらにプレフィックスを委任（delegate）する DHCPv6-PD（DHCPv6-Prefix Delegation）という仕組みもあります。DHCPv6-PD は、ブロードバンドルータなどの CPE（Customer Premises Equipment）に対して ISP が IPv6 アドレスプレフィックスを割り当てる用途などに使われることがあります（IPv4 では、ISP からユーザに対して IPv4 アドレスを 1 つ割り当てる運用が多かったのですが、IPv6 では ISP からユーザに対して IPv6 アドレスを 1 つ割り当てるのではなく、IPv6 プレフィックスを割り当てる場合がよくあります）。

2.9　1つのネットワークインターフェースに複数のIPv6アドレスが設定される

IPv6 では、1 つのネットワークインターフェースに複数の IPv6 アドレスが設定できます。実際、グローバルユニキャストアドレスが設定される場合には、必ず 2 つ以上の IPv6 アドレスが 1 つのネットワークインターフェースに設定されます。これは、そのグローバルユニキャストアドレスだけでなく、すべてのネットワークインターフェースに必ずリンクローカルユニキャストアドレスが設定されるからです。さらに複数の異なるプレフィックスのグローバルユニキャストアドレスが 1 つのネットワークインターフェースに設定されることもあります。

以下、複数の IPv6 アドレスが設定される例をいくつか紹介します。

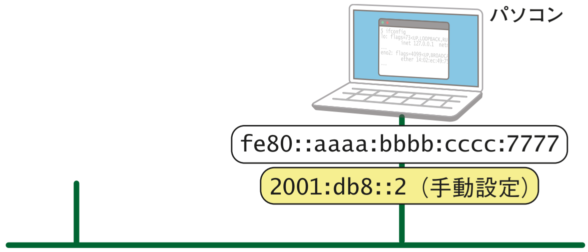
2.9.1　手動設定でグローバルユニキャストアドレスを設定する場合

図 2.8 のように、1 つのネットワークインターフェースに手動でグローバルユニキャストアドレスを設定する場合を考えます。

IPv6 では、すべてのネットワークインターフェースに必ずリンクローカルユニキャストアドレスが設定されます。リンクローカルユニキャストアドレスは、IPv6 が有効になっているネットワークインターフェースが初期化されるときなどに自動で生成されます（7.3 節参照）。

リンクローカルユニキャストアドレスは、その名のとおり同一のリンク内においてのみ通信が可能となる IP アドレスなので、インターネットとの通信にはグローバルユニキャストアドレスも必要です。ネットワークインターフェースに対してグローバルユニキャストアドレスを設

cc BY NC SA

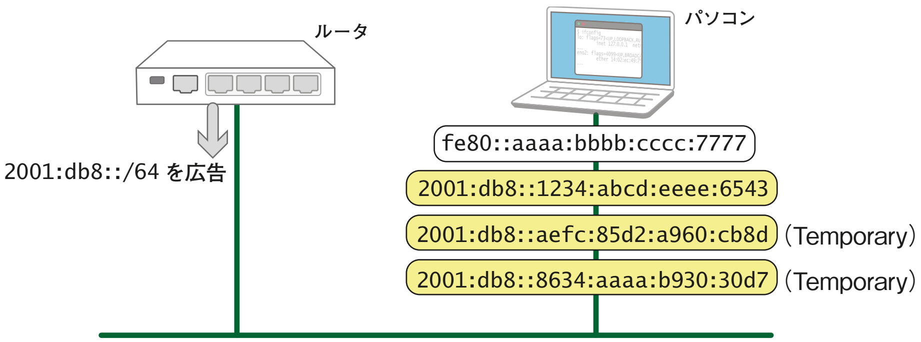


▶ 図2.8　IPv6アドレスが複数設定される例（リンクローカルユニキャストアドレスとグローバルユニキャストアドレス）

定すると、リンクローカルユニキャストアドレスに加えてグローバルユニキャストアドレスを追加する形になります。

2.9.2　IPv6アドレスの自動設定でTemporary IPv6アドレスも利用される場合

ルータなどからの情報を利用してIPv6アドレスが自動設定されるとき、通常のグローバルユニキャストアドレスと同時に**Temporary IPv6アドレス**というものが使われることがあります（図2.9）。Temporary IPv6アドレスは、一定の期間が経過すると無効になるIPv6アドレスです。無効になる前に新たなTemporary IPv6アドレスが自動生成されるので、同じIPv6アドレスを使い続けることを避けることができます。Temporary IPv6アドレスに関しては7.5節で詳しく説明します。



▶ 図2.9　IPv6アドレスが複数設定される例（自動設定とTemporary IPv6アドレス）

2021年現在、ユーザ環境としてパソコンを利用する場合には、Temporary IPv6アドレスがデフォルトで有効な場合が多くなっています。Temporary IPv6アドレスだけでなく、時間経過によって無効にならないグローバルユニキャストアドレス（Stableなアドレス）も設定される場合には、1つのネットワークインターフェースにリンクローカルユニキャストアドレス以外に複数のグローバルユニキャストアドレスが設定されることになります。

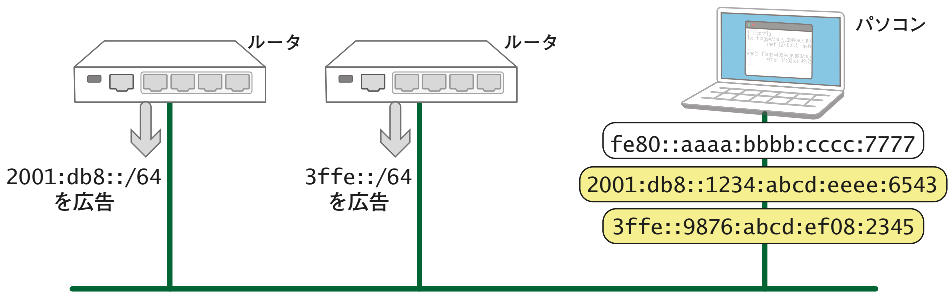
NOTE

設定されるグローバルユニキャストアドレスが1つであれば、上述の手動設定の場合と同様に、リンクローカルユニキャストアドレスとグローバルユニキャストアドレスの2つがIPv6アドレスとして設定されることになります。

2.9.3　マルチホーム環境で複数のプレフィックスが利用される場合

冗長性などを目的にホストやネットワークで複数の接続を維持している環境を**マルチホーム**と呼びます。マルチホーム環境で図2.10のように複数のルータを運用している状況で、ホストがIPv6アドレスの自動設定を利用する場合、2つの異なるプレフィックスを広告するルータからの情報を利用してIPv6アドレスが自動設定されることになります。図2.10の例では、ユーザのパソコンのネットワークインターフェースに対し、次のように複数のIPv6アドレスが設定されます。

- リンクローカルユニキャストアドレス
- グローバルユニキャストアドレス（Stable）：`2001:db8::/64`と`3ffe::/64`
- Temporary IPv6アドレスが使われている場合には、2つのプレフィックスに対応するTemporary IPv6アドレス



▶ 図2.10　IPv6アドレスが複数設定される例（マルチホーム）

2.9.4　IPv6アドレスの選択

IPv6では、1つのネットワークインターフェースに対して複数のIPv6アドレスを設定できます。しかし、パケットのIPv6ヘッダに利用できる宛先および送信元のIPv6アドレスは1つだけです。したがってアプリケーションは、通信の際、複数のIPv6アドレスから1つを選択して利用する必要があります。

どのIPアドレスを選択するのかは、送信元IPv6アドレスだけの話ではありません。たとえば、`www.example.com`のような名前に対してIPv4とIPv6両方のIPアドレスが関連づけられていた場合、そのどちらを優先するのかという判断も必要になります（15.3節参照）。宛先としてIPv6アドレスが選択されたなら、その宛先に送信するときに最適な送信元IPv6アドレスを選択する必要があります。さらに、その宛先に対して最適な次ホップがどの経路かの判断も必要です。環境によっては、名前解決のために利用するキャッシュDNSサーバによって宛先

CC BY NC SA

IPv6アドレスが変わる場合もあるので、それに応じて最適な送信元IPv6アドレスの選択が変わってしまうこともあります。

IPv6アドレスの選択や関連する話題に関しては、13.1節で改めて解説します。

2.10 IPv6とIPv4の違い

本書執筆時点では、家庭用ネットワークやモバイル環境でのIPv6利用が急激に増えているものの、まだIPv4がインターネットにとって重要な要素であり続けています。非常に多くのユーザや組織がIPv4を使い続けており、IPv4をまったく使わずにインターネットを利用するユーザは少数派です。TCP/IPやインターネットの仕組みを学ぶときに、IPv4を前提として学んだ人も多いでしょう。

本書で解説しているIPv6も、IPv4の後継として設計されているので、インターネットというパケット交換網を構成するための基本的な仕組みはIPv4と同様の思想で設計されています。大まかに考えれば、インターネットプロトコルとしての基本的な点については、IPv6もIPv4と変わりません。実際、IPv4と一緒に使われていた他の層のプロトコルの多くは、IPv6でも使えます。IPの上位の層を担うTCPやUDPなどのプロトコルは、IPv6においてもIPv4と同様に利用できます。イーサネットなど、IPの下位の層となるプロトコルも、基本的にはIPv6と一緒に利用できます。

しかし、IPv6には、IPv4によるインターネットが構築されたあとで再設計されたインターネットプロトコルという側面もあります。IPv6とIPv4は、一見するとよく似ているようで、実際には背景となる設計思想からしていろいろ異なるプロトコルです。直接的な互換性もありません。

異なるプロトコルであるからには、相違点の理解が重要です。IPv6を知るには、IPv4との違いを知ることが大切だといえるでしょう。ここまでの説明でも、アドレスの長さをはじめ、IPv6とIPv4のさまざまな違いに触れてきました。すでに説明した事項も含め、ここで改めて両者の違いを一通り整理しておきます。

- **IPアドレス長の違い**

  IPv4アドレスの長さは32ビットだったのに対し、IPv6アドレスの長さは128ビットです。IPv6アドレスのほうがビット数にして4倍の長さがあります。ビット数では4倍の差ですが、表現できるIPアドレスの数にはもっと大きな差があります。1ビット増えるごとに2倍になるので、IPv6のビット数で表現できるIPアドレスの数は、IPv4アドレスの2の96乗倍です。

- **IPアドレスのテキスト表記の違い**

  IPアドレスのテキスト表記の方法も、IPv4とIPv6とでは異なります。具体的には、IPv4アドレスではドット区切りの十進表記、IPv6アドレスではコロン区切りの十六進表記が用いられます。IPv6アドレスとその表記については第3章で詳しく説明します。

- **IPアドレスの構成要素の違い**

  IPv4アドレスは、ネットワーク部とホスト部によって構成されていました。これに対応

するIPv6のユニキャストアドレスは、サブネットプレフィックスとインターフェース識別子によって構成されます。これは単なる名称の違いではなく、IPv6アドレスによって識別するものが、ホストではなくネットワークインターフェースであることを意味しています。IPv6アドレスの構成要素については、2.4節を参照してください。

- アドレスのフォーマットの利用方法が多彩に

  IPv6では、さまざまなプロトコルの機能を実現するために、IPv6アドレスのフォーマットが利用されています。たとえば、IPv6マルチキャストでは、スコープやマルチキャストアドレスの構成がアドレスのフォーマットから判別できるようになっています（11.1節参照）。IPv4/IPv6共存技術として、IPv4アドレスを埋め込んだフォーマットのIPv6アドレスなどもあります。

- ICMPの役割の違い

  IPv4では、IPパケットの転送を実現するうえで、別のプロトコルであるARP（Address Resolution Protocol）やIGMP（Internet Group Management Protocol）などが重要な役割を果たしています。IPv6では、これらの別々のプロトコルが担っていた機能が「近隣探索プロトコル」へと統合されました。IPv6の近隣探索プロトコルでは、従来のIPv4ではエラー通知などに使われているICMP（Internet Control Message Protocol）を拡張したICMPv6が利用されます。そのため、ICMPの重要性がIPv4よりも増しているといえます。ICMPv6については第5章で、近隣探索プロトコルについては第6章で説明します。

- 複数のIPアドレスを1つのネットワークインターフェースに

  IPv4では、1つのネットワークインターフェースに設定するIPアドレスは1つだけでした。これに対し、IPv6では、1つのネットワークインターフェースに対して複数のIPアドレスを設定できることが仕様で明記されています。近隣探索やIPv6アドレスの自動設定の仕様でも、複数のIPv6アドレスをネットワークインターフェースに設定可能であることが前提にされています。また、複数のIPv6アドレスを同時に設定可能であるという特徴を使って、サイト全体で利用されているIPv6アドレスを変更する仕組み（サイトリナンバリング）もあり、大きな議題となっています。サイトリナンバリングについては13.3節で説明します。

  1つのネットワークインターフェースに複数のアドレスが設定されるということは、ネットワークプレフィックスがまったく異なるアドレス宛に、ルータを介さずにパケットを送信できる場合があるということです。つまり、IPv4ではIPv4アドレスとネットマスクを見てon-linkかどうか（同一のリンクに接続しているかどうか）を判定できましたが、IPv6ではネットワークプレフィックスを見てもon-linkかどうかがわからないということです。IPv6では、on-linkかoff-linkかを判断するための仕組みが近隣探索の仕様に関係します。そのため、IPv6におけるon-linkとoff-linkについては、近隣探索プロトコルについて説明する第6章の中の6.6節で説明します。

- IPアドレスの設定、運用方法の違い

  IPv4では、各ホストに対してアドレスを自動的に設定しようと思ったら、IPとは異な

cc BY NC SA

るプロトコルとして定義されているDHCPを利用するしかありませんでした。IPv6では、DHCPv6というプロトコルもありますが、アドレス自動設定のための仕組み（SLAAC）がIPv6プロトコルの一部として定義されています。SLAACをはじめとするIPv6アドレスの自動設定については、第7章で詳しく説明します。

IPv4では、ISPから一般ユーザに割り当てられるIPv4アドレスは1つで、それを家庭内やオフィス内で必要に応じてNATを使って運用するのが一般的でした。IPv6では、ISPからアドレスが1つ割り当てられるのではなく、ネットワークプレフィックスを割り当てられるのが一般的です。そのためIPv6には、ISPなどがネットワークプレフィックスを割り当てるための仕組みとして、DHCPv6-PD（8.7節参照）があります。これもIPv4との大きな違いです。

- ヘッダフォーマットの違い

IPv4とIPv6とではヘッダの仕様が大きく異なります。特に大きいのは、ヘッダ全体の長さが可変だったIPv4に対し、IPv6ヘッダの長さは40バイトに固定されている点です。

IPv4ヘッダが可変長なのは、プロトコルの機能を拡張するために利用されるOptionフィールドが可変長だからですが、IPv6ヘッダにはこのようなフィールドが存在しません。その代わりに、IPv6では、IPv6拡張ヘッダという仕組みが用意されています。

IPv4ヘッダとIPv6ヘッダでは、Optionフィールドの有無以外にも異なる点がたくさんあります。IPv6ヘッダ、IPv6拡張ヘッダ、IPv4とIPv6とのヘッダとの違いについては、第4章で詳しく説明します。

- フラグメンテーションの違い

IPv4では、IPv4ヘッダに含まれるフィールドを利用してIPパケットのフラグメンテーションを実現していました。IPv6ヘッダにはフラグメンテーションのための情報を格納するフィールドはなく、IPv6拡張ヘッダを利用してフラグメンテーションを行います。IPv6におけるフラグメンテーションについては第9章で説明します。

経路の途中でのフラグメンテーションが起きないという点も、IPv6とIPv4の大きな違いの1つです。IPv4では、IPパケットが経路の途中で利用されるデータリンク層のプロトコルの都合でフラグメンテーションが起きる場合がありました。

経路の途中でIPv6パケットのフラグメンテーションができないということは、IPv6ではPath MTU Discoveryが重要になります。Path MTU Discoveryについては第10章で説明します。

- パケットサイズに関する違い

IPv6とIPv4では、パケットサイズに関する要求で異なる部分があります。IPv6の基本仕様を規定するRFC 8200には、“IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater”（「IPv6はインターネットにおけるすべてのリンクが1280オクテットもしくはそれ以上のMTUであることを要求する」）と書かれています。つまりRFC 8200では、MTUが設定可能なリンクでは最低でも1280オクテットの値を設定することを必須としています。

CC BY NC SA

これに対しIPv4の基本仕様を規定するRFC 791[†13]では、IPv4ヘッダのTotal Lengthフィールドの仕様として、“All hosts must be prepared to accept datagrams of up to 576 octets”（「最低でも576オクテットのデータグラムを受け取れるように準備していなければならない」）とあります（IPv4のTotal Lengthフィールドの仕様では「MTU」という表現は使われていません）。そのうえで、受信相手がそれ以上のサイズのデータグラムを受け取れるという保証があるのであれば、送信側は576オクテット以上のパケットを送信することを推奨（recommended）しています。

- **ブロードキャストからマルチキャストへ**

  IPv6ではブロードキャストが廃止されました。IPv4ではブロードキャストで行われていたことがマルチキャストで行われるようになります。マルチキャストについては第11章で説明します。

2.11 IPv6対応とは

IPv6はIPv4と同じネットワーク層のプロトコルであり、ネットワークを越えてパケットを配送するという点では、インターネットにおいてIPv4と同じ機能を担うプロトコルです。しかし、IPv4の制約を見据えて新たに生み出されたIPv6には、前節で挙げたようにIPv4との違いもたくさんあります。単純にIPv4よりアドレスが長くなっただけでなく、IPv4とは似て非なるネットワーク層プロトコルであるIPv6は、IPv4と直接的な互換性がありません。

では、長らくIPv4を中心に運用されていたインターネットでIPv6を導入するにあたり、どのようなことを考えなければならないでしょうか。前節に挙げたようなIPv4とIPv6のさまざまな違いを理解することは、単にプロトコル仕様の把握という面だけでなく、ネットワークをどのように運用するかという面でも意味があります。特に、本書執筆時点のインターネットでは既存のIPv4が使われ続けているので、IPv6のみのネットワーク運用について考えるだけでなく、同じインターネット上でIPv4と並行してIPv6が利用される**混在環境**での運用を考慮しなければなりません。たとえば、「両方のプロトコルの通信を同じように扱うのにファイアウォールの設定方法が異なる」といった影響が出てくるでしょう。

しかも、そのような混在環境はかなり長い期間にわたって継続することが予想されます。その期間に世界中のネットワークで要求されるのは、IPv4中心のインターネットで、IPv6という新しいネットワーク層プロトコルを利用できるようにすることです。つまり、**IPv6への移行**よりも**IPv6対応**が求められるのです。

- **IPv6への移行**：IPv4を捨ててIPv6に完全に移ること
- **IPv6対応**：IPv4を利用するユーザが多い状況で、それまで対応していなかったIPv6という新しいプロトコルに対応すること

この節では、これら2つのうち「IPv6対応」について具体的に考えます。一言でIPv6対応といっても、それが意味するところは状況によってさまざまです。インターネットのどこかを調

[†13] RFC 791：J. Postel, “Internet Protocol”, 1981年9月

CC BY NC SA

整すれば「IPv6対応」ができるわけではなく、いろいろなところで、いろいろな形の「IPv6対応」が必要になります。

IPv6対応として具体的に必要になるのは、利用する機器がIPv6対応になること、利用しているネットワークがIPv6対応になること、利用しているネットワークがIPv6インターネットと通信可能になることです。ここでは、これらのIPv6対応について、以下の4つの視点に分けて考えていきます。

- ネットワークのIPv6対応
- ユーザのIPv6対応
- サーバのIPv6対応
- アプリケーションのIPv6対応

2.11.1 ネットワークのIPv6対応

IPv6による通信を行うには、ネットワークのIPv6対応が必要です。ここで注意してほしいのは、通信プロトコルとしてIPv6を利用することと、IPv6で作られたインターネット（IPv6インターネット）を通じて世界中のサーバと通信できる環境を整えることは、一見同じように思えて実際には互いに独立した話であるという点です。

NOTE

インターネットとの通信ができない閉域網において、ネットワーク層（レイヤー3）の通信プロトコルとしてIPv6が使われることもあります。そのような場合を指して**IPv6ネットワーク**と表現し、IPv6で作られた**IPv6インターネット**とは区別する場合もあります。ただし、IPv6インターネットとの通信を実現することを指して「IPv6ネットワーク」と表現する場合も少なくありません。

ネットワークがIPv6対応しているだけでは、IPv6インターネットとの通信はできません。ネットワークがIPv6インターネットと接続されていることも必要です。IPv6インターネットに接続する方法としては、IPv6インターネットに直接接続する方法と、IPv4インターネットを通じてIPv6インターネットに接続する方法があります。IPv4インターネットを通じてIPv6インターネットに接続する方法としては、IPv4インターネットにトンネルを構築してIPv6インターネットに接続する6rdなど、さまざまなプロトコルがあります（詳細は第V部で解説します）。

これとは逆に、DS-LiteやMAPなど、IPv6インターネットを利用してIPv4インターネットへの接続を提供する手法もあります。IPv6インターネットをどのようにユーザに提供するのかが、IPv4インターネットをどのようにユーザに提供するのかという話に影響を与える場合もあるのです。これらの話題については第24章や付録Aで紹介します。

2.11.2 ユーザのIPv6対応

ユーザに関するIPv6対応では以下のような論点があります。

- （1）ユーザが利用している機器がIPv6に対応しているか

CC BY NC SA

- （2）ユーザが利用している機器でIPv6が有効になっているか
- （3）ユーザが利用しているネットワーク機器がIPv6に対応しているか
- （4）ユーザが契約している回線でIPv6インターネットへの接続サービスが提供されており、ユーザがそのサービスを契約しているか

（1）の「ユーザが利用している機器がIPv6に対応しているか」については、本書執筆時点で一般的なユーザが利用しているパソコンで使われるOSはすべてIPv6対応にしています。一方、モバイル機器に関してはIPv6に対応していないものがまだ残っています。

（2）の「ユーザが利用している機器の設定においてIPv6を有効にしているかどうか」も、ユーザのIPv6対応を考えるうえでは大きなポイントです。その際に影響として大きいのは、設定が可能かどうかよりも、むしろ初期設定（デフォルト設定）がどうなっているかです。ユーザが実際にIPv6を利用するかどうかは、機器を入手した際の初期設定でIPv6が有効となっているか否かによってほぼ決まります。

（3）の「ユーザが利用しているネットワーク機器がIPv6に対応しているか」も重要な要素です。一般家庭におけるインターネットへの出入り口部分や、家庭内LANがIPv6対応していなければ、そのユーザはIPv6を利用できません。一般家庭では、IPv6に対応していない古いルータが使われ続けていることも少なくありません。

最後に、（4）の「ユーザが契約している回線でIPv6インターネットへの接続サービスが提供されており、ユーザがそのサービスを契約しているか」という点も問題になります。この部分でIPv6対応を実現するには、IPv4とIPv6を両方とも同じネットワークで提供する方法、IPv4をIPv6経由で提供する方法、IPv6をIPv4経由で提供する方法など、さまざまな手法があります。さまざまな手法が存在するのは、インターネット接続性を提供する事業者がどういったネットワークを運用しているのかによって利用可能な方法が異なるからです。第V部で紹介する技術のいくつかは、ユーザに対してインターネット接続性を提供するための技術だといえます。

2.11.3　アプリケーションのIPv6対応

ネットワークがIPv6対応しており、ユーザのIPv6対応が万全な状況であっても、使われるアプリケーションがIPv4でのみ通信可能な状態であればIPv6が使われることはありません。OSがIPv6対応しているからといって、そのOS上で使われるアプリケーションがすべてIPv6対応しているというわけではないのです。アプリケーションではアプリケーションのためのIPv6対応が必要になります。そもそも、IPv4を使って通信を行うのか、それともIPv6を使って通信を行うのかを判断するのは、個々のアプリケーションです。

アプリケーションをIPv6対応にするためのプログラミングに関しては、第15章で説明します。アプリケーションにおけるIPv6対応の注意点などがまとめられたRFC 4038[†14]では、次のような点について言及されています。

†14　RFC 4038：M-K. Shin, Y-G. Hong, J. Hagino, P. Savola, E. M. Castro, “Application Aspects of IPv6 Transition”, 2005年3月

CC BY NC SA

- 従来のIPv4に対応しつつ完全にIPv6対応バージョンを提供するのが困難な場合もある
- IPアドレスの表記方法がIPv4とは違うので、IPアドレスをパースする際には注意が必要である
- 送信元IPv6アドレスの選択が必要な場合もある（13.1.1項参照）
- マルチキャストを使うアプリケーションは、IPv4とIPv6におけるマルチキャストの違いに注意する
- プログラミングではAPI（Application Programming Interface）などの違いに注意する

このほか、接続相手のIPアドレスを保持するデータがある場合、IPv4アドレスとIPv6アドレスで32ビットと128ビットという違いがあります。また、そのデータがIPv4アドレスであるかIPv6であるかを判別するために別のデータが必要になる場合もあります。

2.11.4 サーバのIPv6対応

サーバのIPv6対応に関しては、コンテンツをIPv6に移行する方法がRFC 6589[†15]で、コンテンツプロバイダやアプリケーションサービスプロバイダのためのIPv6導入方法がRFC 6883[†16]で、それぞれ解説されています。

サーバの名前に対応するIPv6アドレスをDNSサーバに登録するときは、AAAAレコードという形で登録されます（DNSやAAAAレコードについては2.12.2項と第16章で簡単に説明します）。RFC 6883には、「知られている名前に対してAAAAレコードがDNSに登録されるとすぐに、そのAAAAレコードに対応するサーバはIPv6トラフィックを受け取り始める」[†17]とあります。IPv6で通信可能なサーバを用意することで、IPv6インターネットに接続されたユーザとサーバとが通信できる状態になります。しかし、Webをはじめとする多くのアプリケーションは、通常、名前解決を行ったうえで通信を行います。もちろん、IPアドレスを最初から指定して通信することもありえますが、一般的なサービスでは名前解決を行い、その結果得られるIPアドレスを使って通信します。そのため、IPv6インターネットに接続されたサーバに対する実トラフィックも、DNSにAAAAレコードが登録された段階で、初めて本格的に発生することになります。サーバに対するIPv6での実トラフィックが発生するのは、DNSサーバにAAAAレコードが登録されたあとになることが多いのです。

ここで注意が必要なのは、接続にIPv4を使うのか、それともIPv6を使うのかは、通信を開始するクライアント側アプリケーションが判断するという点です。サーバ運用者側は、DNSにAAAAレコードを登録することで、その判断のための選択肢を提供することになります。

サーバ側でIPv6対応が必要なのは、実際にIPv6で通信しながらサービスを提供しているサーバアプリケーションだけでない場合もあります。たとえば、サーバアプリケーションのログにIPv6アドレスが含まれており、別のプログラムがそのIPv6アドレスを解釈する必要があることもあります。Webサーバで稼働するアプリケーションがSQLを利用しており、そこで

---

†15 RFC 6589：J. Livingood, “Considerations for Transitioning Content to IPv6”, 2012年4月

†16 RFC 6883：B. Carpenter, S. Jiang, “IPv6 Guidance for Internet Content Providers and Application Service Providers”, 2013年3月

†17 原文は、“It must be understood that as soon as a AAAA record for a well-known name is published in the DNS, the corresponding server will start to receive IPv6 traffic.”

CC BY NC SA

利用しているテーブルにIPv4アドレス専用のカラムが存在していることも考えられます。

サーバを守るためのセキュリティに関しても注意が必要です。IPv6が関連するセキュリティに関しては、第14章で説明します。

2.12　IPv6インターネットとIPv4インターネットを同時に使う

IPv4は、インターネットの誕生時点から世界中に普及した現在に至るまで使われてきたプロトコルです。一方のIPv6は、これから普及していくプロトコルです。今後しばらくは、互いに直接的な互換性がないプロトコルを利用する「IPv4インターネット」と「IPv6インターネット」という2種類のインターネットが、利用者から見れば1つのインターネットとして同時に運用されている状態が続くことになります。

この節では、これら2種類のインターネットが同時に運用される状況を整理し、その状態で1つのインターネットを実現するうえで重要な要素となる名前解決について説明します。

2.12.1　IPv6とIPv4のデュアルスタック

ある特定のプロトコルを扱うためのソフトウェアの実装を**プロトコルスタック**と呼びます。IPv4とIPv6は異なるプロトコルなので、プロトコルスタックとしても別です。まったく別々の2つのプロトコルスタックを使える状態を**デュアルスタック**と呼びます。

1.2節ではインターネットの機能を説明するのに階層構造を利用しましたが、プロトコルスタックもこの階層構造で考えることができます。IPv4だけのインターネットを実現していたプロトコルスタックは、図2.11のような階層構造として表現できます。図2.11からわかるように、IPv4が位置するネットワーク層だけは単一のプロトコルで、それ以外の層にはすべて複数のプロトコルが存在しています。IPv6が登場する前はIPv4一択だったので、インターネットは図2.11のようなプロトコルスタックだったのです。



▶ 図2.11　IPv4のみの場合のプロトコルスタック

しかし、IPv4よりアドレス空間が広いIPv6が利用されるようになった現在では、単一のプロトコルが前提であったネットワーク層に2つのプロトコルが存在する状態になっています。図2.12のようなプロトコルスタックが、ネットワーク層にIPv4とIPv6の両方が存在する、現在のインターネットの状態です。

このように、IPv4とIPv6が並行して存在するデュアルスタックのインターネットでは、互



▶ 図2.12　IPv4とIPv6の両方が存在する場合

換性のない2つのネットワーク層プロトコルが利用されます。1つが前提であったネットワーク層が、2つの別のプロトコルが共存するデュアルスタック環境になることを意識しなければならない場面は、サーバやネットワークの管理者、通信が関連するプログラムを書くプログラマだけでなく、一般のインターネット利用者にとっても増えていくと考えられます。

2.12.2　DNSの役割

ネットワーク層がIPv4とIPv6のデュアルスタックになっても、ネットワークを利用する側の視点で見れば、「インターネットは1つ」でなければ困ります。IPv4とIPv6を並行して利用できる現在の状況は、「ネットワーク層プロトコルで見れば2つの別のプロトコルによって実現される2つのインターネットだが、インターネットとしてはあくまでも1つ」という、非常にややこしい状況にあるといえるでしょう。

現在のインターネットでは、この「2つであるが1つ」という状態を仮想的に実現するために、**DNS**（Domain Name System）を利用しています。DNSは、インターネットでの通信に必要な通信相手のIPアドレスを名前から得るための**名前解決**に利用されている仕組みです。このDNSがIPv4とIPv6が混在するインターネットの大きなポイントです。2つの異なるインターネットプロトコルで「1つのインターネット」を実現するために名前空間を共有するということは、DNSでIPv4とIPv6の両方に対応する**名前**を扱うことを意味します。現在のインターネットは、名前に関してはIPv4とIPv6をあえて混ぜた状態で設計されているといえるでしょう。

DNSでは、ドメイン名という名前からIPアドレスを解決するために、レコードと呼ばれる仕組みを利用しています。レコードは、IPv4アドレス用とIPv6アドレス用とを別々に設定できます。たとえば、「`www.example.com`」という1つの名前に対し、IPv4とIPv6の両方のIPアドレスを登録できるようになっています。IPv4アドレス用は**Aレコード**、IPv6アドレス用は**AAAAレコード**（「クアッド・エー」と読みます）と呼びます。

なぜ、1つの名前に対して2つのIPアドレスを登録できることが重要なのでしょうか？ Webにおける通信を例に考えてみましょう。「`http://www.example.com/`」というURLを持つサーバと通信をするときにユーザがアクセスしたいのは、通信がIPv4によるか、それともIPv6によるかにかかわらず、「`www.example.com`」が指し示すWebページです。仮に、IPv4とIPv6とで名前空間がまったく異なるものだったとしたら、IPv4で「`example.com`」というドメイ

CC BY NC SA

ン名の登録を行っている組織と、IPv6 で「example.com」というドメイン名の登録を行っている組織が違う可能性もあります。もしそうなっていたら、IPv4 で通信する場合と IPv6 で通信する場合とで、同じ「http://www.example.com」という URL でアクセスできる Web ページも別物になってしまいます。それではユーザが混乱してしまうでしょう。

名前空間のための DNS の **Root**（根っこ）は、「1 つ」であることが重要です（RFC 2826[†18]）。IPv4 インターネットと IPv6 インターネットがまったく異なるネットワークであるのは、あくまで通信プロトコル上の話であり、名前空間を含めたいわゆる「インターネット」としては「1 つ」なのです。

2.12.3 IPv4 と IPv6 のどちらを使うのか判断するのはユーザ側

IPv4 だけを利用する場合には IPv4 アドレスに対する名前解決が行われ、IPv6 だけを利用する場合には IPv6 アドレスだけの名前解決が行われます。IPv4 と IPv6 の両方の利用を試みる場合には、IPv4 と IPv6 の両方に対する名前解決が行われます。ここで重要になるのが、「IPv4 と IPv6 のどちらを使って通信するのかを誰がどのように判断するか」という点です。

結論から言うと、この判断をするのはユーザ側です。そのため、ユーザは、名前に関して IPv4 と IPv6 の問い合わせを別々に行う必要があります。別々の問い合わせを行うということは、ユーザ側が明示的に IPv4 と IPv6 の両方に関して名前解決をしたいと DNS サーバに問い合わせていることになります。このとき DNS サーバは、個々の問い合わせに回答しているだけです。DNS サーバが「あなたは IPv4 を使いなさい」とか「あなたは IPv6 を使いなさい」などという指示を出しているわけではありません。

IPv4 と IPv6 の両方で運用されている Web サーバを例にして考えましょう。サーバ側は、たとえ IPv4 と IPv6 の両方で TCP ソケットを使って接続されるのを待っていたとしても、ユーザ側が IPv6 での接続を行ってくれなければユーザと IPv6 による通信はできません。たとえば、「www.example.com」というドメイン名の Web サイトが、IPv4 と IPv6 の両方で運用されていたとします。この Web サーバに対して、ユーザが Web ブラウザで接続する場合には、まず DNS で「www.example.com」の名前解決を試みる必要があります。その際には、IPv4 の A レコードと、IPv6 の AAAA レコードの両方に対して、名前解決を試みます。もし両方ともに結果が返ってきた場合、IPv4 で接続するのか、それとも IPv6 で接続するのかを判断するのは、Web ブラウザです。

このように、接続に IPv4 を使うのか、それとも IPv6 を使うのかを判断するのは、個々のアプリケーションの仕事です。他の環境がすべて同じであったとしても、IPv4 と IPv6 のどちらを使うべきかは、アプリケーションごとに異なる可能性もあります。アプリケーションプログラマは、IPv4 を使うのか、それとも IPv6 を使うのかを判断するコードを、アプリケーションごとに書く必要があります。

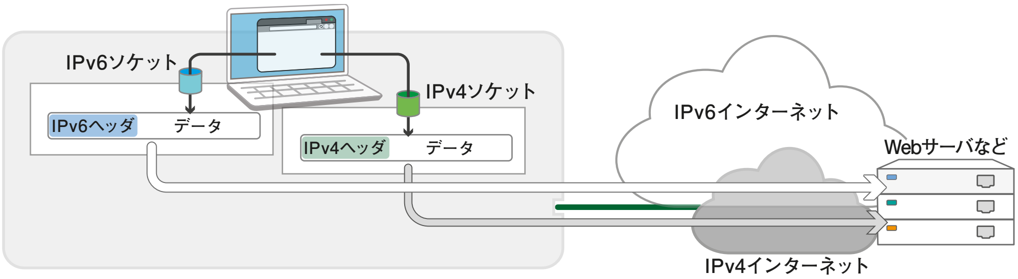
かつては IPv6 で運用されているサーバが非常に少なかったので、IPv4 と IPv6 のどちらで通信を行うのかをアプリケーションプログラマが考える必要性は、事実上、皆無でした。TCP で通信を行うには、IPv4 アドレスを名前解決で得たのちに TCP で接続するだけだったのです。

[†18] RFC 2826：Internet Architecture Board, “IAB Technical Comment on the Unique DNS Root”, 2000 年 5 月

CC BY NC SA

しかし、本書執筆時点では、IPv4 を使うのか、それとも IPv6 を使うのかを判断するプログラムを書くことが強く推奨されるようになっています。

とはいえ、IPv4 と IPv6 のそれぞれで TCP の接続が成功するかどうかは、実際に接続を試みないとわかりません。そのため、図 2.13 のように、IPv4 と IPv6 の両方を同時に接続してしまい、先に成功したほうの接続を使うという方法もあります[†19]。



▶ 図 2.13　IPv4 と IPv6 の両方で接続を試みる

IPv6 がさらに普及し、IPv4 の利用が少なくなれば、今度は IPv6 だけで接続すればよくなるので、再び IPv4 と IPv6 のどちらのプロトコルを使うのかを判断する必要がなくなるかもしれません。その頃には本節のような説明も不要になり、「昔は IPv4 というものもありました」という、軽い紹介だけで済むでしょう。

2.13　IPv4 と IPv6 の共存技術

IPv4 と IPv6 の間には、直接的な互換性はありません。しかし、IPv4 と IPv6 を運用するための物理的な回線などが共用可能であることを活用するための技術も考案されています。そういった技術は**移行技術**や**共存技術**と呼ばれています。

本書では、IPv4 と IPv6 の共存技術を次の 3 種類に分類しています[†20]。

1. IPv6 ネットワーク同士をつなぐ IPv4 トンネルを実現する技術
2. IPv4 ネットワーク同士をつなぐ IPv6 トンネルを実現する技術
3. IPv4 ネットワークに接続された機器と IPv6 ネットワークに接続された機器の間で通信を行えるようにする技術

上記のうち、1 と 2 のトンネル技術では、IP パケットが IP パケットでカプセル化して運ばれます。IPv4 インターネットと IPv6 インターネットを相互につなげるのではなく、あくまでペイロードとなる IP パケットをカプセル化して配送したうえで、トンネルを抜けた段階で脱カプセル化します。一方、3 の技術では、トンネル技術と違って IPv4 パケットと IPv6 パケットの変換を行います。

以下、上記のそれぞれについて概要を紹介します。

†19　15.3 節で説明するように、多少 IPv6 のほうが有利になるようなタイミングでの接続が行われることもあります。

†20　後の章では、IPv4 ネットワーク、IPv4 インターネット、IPv6 ネットワーク、IPv6 インターネットの 4 種類で分類しますが、ここではひとまず閉じたネットワークとインターネットの違いに関しては割愛した分類にしています。

BY NC SA

2.13.1 IPv6 ネットワーク同士をつなぐ IPv4 トンネルを実現する技術

初期に作られたのは、IPv4 を経由して IPv6 インターネットに接続するための技術でした。IPv4 トンネルで IPv6 パケットを運ぶ技術です。

IPv6 の標準が作られた当初は、IPv6 インターネットは非常に小規模でした。 その当時、IPv4 インターネットが世界中に急速に拡大する一方で、IPv6 インターネットは「島」のように点在する IPv6 ネットワークをつないだような状態でした。それぞれの「IPv6 島」をつなぐのに使われていたのが IPv4 トンネルです。IPv4 パケットの中に IPv6 パケットをカプセル化し、IPv4 トンネルを経由して IPv6 ネットワークへとつなぐという手法です。いまでは利用が推奨されないものもありますが、IPv4 トンネルを経由して IPv6 インターネットへと接続する技術として、6to4、Teredo、ISATAP、6rd などの技術があります。これらの技術は、IPv6 が普及するまでは IPv4 に頼りつつ IPv6 インターネットへの接続を行うという思想から、「IPv6 への移行技術」と呼ばれていました。

2000 年代は、これらの技術が IPv6 に関連して話題となることが多かったのですが、当時と比べて現在では IPv6 インターネットに接続できる商用サービスが一般的になっています。そのため、2021 年現在ではこれらの技術を利用する必要性は減少しています。

2.13.2 IPv4 ネットワーク同士をつなぐ IPv6 トンネルを実現する技術

IPv6 インターネットへと接続する環境が整っていくと、IPv4 と IPv6 を同時に運用するときにコストを削減する方法をどのように実現するのかという議論が活発になりました。その代表的なものとして挙げられるのが、米国 Comcast で使われている DS-Lite（Dual Stack Lite）です。DS-Lite は、基幹ネットワークを IPv6 のみで構成しつつ、IPv4 のための IPv6 トンネルを使うことで、ユーザに対して IPv4 と IPv6 の両方を提供できるというものです。同様に IPv4 パケットを運ぶために IPv6 トンネルが利用する技術としては、DS-Lite のほかに MAP-E があります。DS-Lite については 24.1 節で、MAP-E については 24.4.1 項でそれぞれ解説します。

日本では、NTT フレッツ網における IPv4 over IPv6 技術として、DS-Lite や MAP-E が話題になることがあります。NTT フレッツ網に関しては付録 A で説明します。

2.13.3 IPv4 ネットワークに接続された機器と IPv6 ネットワークに接続された機器の間で通信を行えるようにする技術

IPv4 ネットワークに接続された機器と IPv6 ネットワークに接続された機器の間で通信を行えるようにする技術としては、NAT64 と DNS64 があります。NAT64 と DNS64 を利用して IPv4 と IPv6 を変換することにより、ユーザ環境を IPv6 シングルスタックにしつつ、IPv6 ネットワークを経由して IPv4 インターネットとの通信を実現できます[†21]。ただし、NAT64 は NAT の一種であることから、IPv4 NAT 同様にさまざまな制約があります。

NAT64 と DNS64 を利用する商用サービスも徐々に登場しつつあります。2021 年時点では、BIGLOBE と楽天が NAT64 と DNS64 によるインターネット接続サービスを提供しています。NTT docomo でも 2022 年春から NAT64 と DNS64 によるインターネット接続サービスを開始

[†21] NAT64 と DNS64 を利用しつつ、ユーザ環境は IPv6 と IPv4 のデュアルスタックにすることも可能です。

CC BY NC SA

すると発表しています。

NAT64とDNS64については23.3節で解説します。

CC BY NC SA

[書籍版では空白のページです]

BY NC SA

第II部

IPv6プロトコルとその周辺技術

ここでは、IPv6および周辺機能の基本的な仕様および関連するトピックについて説明します。主な話題は次のようなものです。

- IPv6アドレス体系
- IPv6ヘッダとIPv6拡張ヘッダ
- ICMPv6と近隣探索プロトコル
- IPv6アドレスの自動設定（SLAACおよびDHCPv6）
- IPフラグメンテーションとPath MTU Discovery
- マルチキャストとエニーキャスト
- マルチアドレスとマルチプレフィックス
- セキュリティとアプリケーションのIPv6対応

なお、実際のIPv6ノードで要求されるさまざまな機能については、InformationalなRFCとして、2019年2月に発行されたRFC 8504“IPv6 Node Requirements”でまとめられています。家庭用のIPv6対応ルータなどに求められる機能を紹介したRFCとしては、2013年11月に発行されたRFC 7084“Basic Requirements for IPv6 Customer Edge Routers”があります。

[書籍版では空白のページです]

CC BY NC SA

第3章

IPv6アドレス体系

IPv6には、RFC 4291[†1]でまとめられている「IPv6アドレス体系」(IPv6 addressing architecture)という概念があります。IPv4には「IPアドレス体系」のような体裁でまとめられたRFCはなかったので、RFC 4291の存在はIPv6の大きな特徴のひとつといえます。

IPv6アドレス体系という概念がIPv6に存在するのは、IPアドレスの長さが128ビットに増えたことが要因です。IPアドレスの長さが32ビットから128ビットに増えることの意味は、「利用可能なIPアドレスの総数が増える」というだけではありません。「ビットの並び方に意味を持たせる方法がIPv4よりも多彩になる」という側面もあります。そのため、それらを体系的に整理するRFCが必要になったのです。

この章では、主にRFC 4291でまとめられているIPv6アドレス体系を中心に解説しています。また、IPv6アドレスが実際のネットワークでどのように使われるかについて説明します。

3.1 IPv6アドレスの種類

RFC 4291では、IPv6アドレスの基本的な分類を下記のように定義しています。

アドレスの種類	IPv6アドレス
未定義アドレス	`::/128`
ループバックアドレス	`::1/128`
マルチキャストアドレス	`ff00::/8`
リンクローカルユニキャストアドレス	`fe80::/10`
グローバルユニキャストアドレス	上記以外のすべてのアドレス

- 未定義アドレス(`::/128`)

  RFC 4291では、128ビットすべてが0となるIPv6アドレスを未定義アドレス(Unspecified Address)としています。未定義アドレスは、IPv6アドレスが存在しないことを示すために予約された値です。

[†1] RFC 4291:R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", 2006年2月

CC BY NC SA

このIPv6アドレスをノードに対して割り当てることはできません。また、宛先IPv6アドレスとして利用することもできません。

未定義アドレスは、たとえばIPv6アドレスの割り当てを受ける前段階のノードが送信元IPv6アドレスとして利用します。ただし、送信元IPv6アドレスが「`::/128`」のパケットをルータが転送することは禁止されています。

> **NOTE**
> ルーティングプロトコルでデフォルト経路を表現するために利用されるのは、IPv6では「`::/0`」、IPv4では「`0.0.0.0/0`」です。それぞれプレフィックス長が0です。

- ループバックアドレス（`::1/128`）

  ループバックアドレスは、自分自身を示すIPアドレスです。IPv4における`127.0.0.1`に相当するアドレスです。

  ループバックアドレスは、ノード内のみで利用可能です。ループバックアドレスを送信元もしくは宛先アドレスとしてノード外に送信することは禁止されています。ループバックアドレスを宛先アドレスとしたパケットを物理インターフェースから受け取った場合、そのパケットは破棄されます。

- マルチキャストアドレス（`ff00::/8`）

  いくつかのノード宛にまとめてパケットを送信するときに使えるアドレスです。先頭8ビットが1のIPv6アドレスは、すべてマルチキャスト用として確保されています。IPv6では、プロトコルのさまざまな機能を実現するために、マルチキャストが非常に効果的に利用されています。

  ユニキャストIPv6同様に、IANAによって予約済みのマルチキャストIPv6アドレスもあります。よく使われるマルチキャストIPv6アドレスとして、以下の2種類が挙げられます（RFC 4291に記載）。

  - 同一リンク内の全ノード

    `ff02::1`（全ノードアドレス、All Nodes Address）

  - 同一リンク内の全ルータ

    `ff02::2`（全ルータアドレス、All Routers Address）

  マルチキャストアドレスについては、11.1節で詳しく説明します。

- リンクローカルユニキャストアドレス（`fe80::/10`）

  同一のリンク内でのみ利用可能なアドレスです。詳しくは3.5節で説明します。

- グローバルユニキャストアドレス

  上記以外のすべてのアドレスが、グローバルユニキャストアドレスです。

上記のほかにも、さまざまな特殊用途のIPv6アドレスがあります。特殊用途のIPv6アドレ

(cc) BY NC SA

スは、RFC 6890[†2]（BCP 153）としてまとめられています。

3.2 IPv6アドレス空間の使い方はIANAが管理している

IPv6 アドレスの使い方は、IPv4 同様に IANA が管理しており、下記の URL で参照できます[†3]。

- https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml

IANA で割り当てられている IPv6 アドレス空間の一部を表 3.1 に抜粋します。

▶ 表 3.1 IANA で予約されている IPv6 アドレス

アドレスブロック	RFC	説明
0000::/8	3513、4291	特殊用途のために予約されている IPv6 アドレスブロックを含む。表 3.2 参照
0100::/8	3513、4291	0100::/64 は Discard-Only Address Block として予約されている（RFC 6666）。表 3.2 参照
0200::/7	4048	RFC 3513 にて、“reserved for NSAP Allocation” とされ RFC 1888 が参照されていたが RFC 4048 によって廃止
2000::/3	-	IPv6 アドレス全体のうちマルチキャスト ff00::/8 以外はすべてユニキャストアドレスとされているが、IANA および RIR などからの割り振りは 2000::/3 に限定されている。他の IPv6 アドレス空間は “Reserved by IETF”
fc00::/7	4193	ULA（Unique Local Unicast）。3.8 節参照
fe80::/10	3513、4291	リンクローカルアドレス。IANA のサイトでは “Link-Scoped Unicast” と表現されている。IPv6 アドレスのスコープに関しては 3.3 節、リンクローカルアドレスに関しては 3.5 節をそれぞれ参照
fec0::/10	3879	Site-Local scoped address prefix として予約されていたが RFC 3879 によって廃止
ff00::/8	3513、4291	マルチキャスト。第 11 章参照

本書執筆時点で IPv6 グローバルユニキャストアドレスとして IANA で予約されているのは、2000::/3 です。IPv6 グローバルユニキャストアドレスの RIR に対する割り振りは、上記とは別の下記の URL でまとめられています。

- https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml

特殊用途のために予約されている IPv6 アドレスは、下記の URL で参照できます。

- https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml

特殊用途のために予約されている IPv6 アドレス空間を表 3.2 に示します。

†2 RFC 6890：M. Cotton, L. Vegoda, R. Bonica, B. Haberman, “Special-Purpose IP Address Registries”, 2013 年 4 月

†3 IANA に関しては 1.4.6 項を参照。

▶ 表3.2　IANAで予約されている特殊用途のIPv6アドレス

アドレスブロック	名称（IANA記載）	RFC	説明
::1/128	Loopback Address	4291	ループバックアドレス
::/128	Unspecified Address	4291	未定義アドレス
::ffff:0:0/96	IPv4-mapped Address	4291	IPv4-Mapped IPv6アドレス。3.9節参照
64:ff9b::/96	IPv4-IPv6 Translat.	6052	23.4節参照
64:ff9b:1::/48	IPv4-IPv6 Translat.	8215	23.4.1項参照
100::/64	Discard-Only Address Block	6666	ブラックホール用IPv6アドレス。14.12節参照
2001::/23	IETF Protocol Assignments	2928	上位16ビットが0x2001でそれに続く7ビットがすべて0となるIPv6アドレスブロックは、RFC 2928のsection 3にてテストや実験のためにIANAに割り当てられている
2001::/32	TEREDO	4380、8190	Teredo。22.2節参照
2001:1::1/128	Port Control Protocol Anycast	7723	Port Control Protocolのエニーキャストアドレスとして、192.0.0.9と共にIANAによって割り当てられている
2001:1::2/128	Traversal Using Relays around NAT Anycast	8155	TURNサーバの自動的な発見。20.3.1項参照
2001:2::/48	Benchmarking	5180	RFC 2544で示されるベンチマーキングの方法論がIPv6のいくつかの機能に対応していないので、その点を補う内容を示すために使われる。RFC 5180のsection 8ではIANAでの割り当てを2001:0200::/48としているが、Errata 1752によると実際に割り当てられたのは2001:2::/48
2001:3::/32	AMT	7450	Automatic Multicast Tunnelingで利用される
2001:4:112::/48	AS112-v6	7535	AS112は、プライベートIPv4アドレスなどのユニークではないIPアドレスの逆引きを行うための仕組み。RFC 7535ではDNAMEを利用したAS112が定義されている
2001:10::/28	Deprecated (previously ORCHID)	4843	Overlay Routable Cryptographic Hash Identifiers（ORCHID）で利用されていた。RFC 4843はRFC 7343によって廃止されている
2001:20::/28	ORCHIDv2	7343	Overlay Routable Cryptographic Hash Identifiers Version 2（ORCHIDv2）で利用
2001:30::/28	Drone Remote ID Protocol Entity Tags (DETs) Prefix	ietf-drip-rid-37	RFC 9153で定義された無人機用プロトコルのDRIP（Drone Remote ID Protocol）で使うHHIT（Hierarchical Host Identity Tag）で要求される
2001:db8::/32	Documentation	3849	例示用IPv6アドレス。3.11節参照
2002::/16	6to4	3056	22.1節参照
2620:4f:8000::/48	Direct Delegation AS112 Service	7534	AS112のノードがBGPでAS112のnameserverが存在していることを示すために利用するIPv6アドレス
fc00::/7	Unique-Local	4193、8190	ULA（Unique Local Unicast）。3.8節参照
fe80::/10	Link-Local Unicast	4291	リンクローカルアドレス。3.5節参照

(cc) BY NC SA

3.3 IPv6アドレスのスコープ

IPv6には、IPv6アドレスの有効範囲を示す**スコープ**という概念があります。IPv6アドレスのスコープはRFC 4007[†4]で定義されています。ユニキャスト（およびエニーキャストアドレス）にはリンクローカルスコープとグローバルスコープの2種類があります。

サイトローカルアドレスが廃止される前は、サイトローカルアドレスもユニキャストおよびエニーキャストのスコープを構成していました。しかし、サイトローカルアドレスが廃止されたため、ユニキャストおよびエニーキャストでのスコープは2種類だけになっています。3.8節で紹介するULA（Unique Local IPv6 Unicast Address）のスコープはグローバルです。

例外として、128ビットすべてが0である未定義アドレス（::）は、どのスコープにも属していません。その理由としてRFC 4007には“because it must never be assigned to any node”（「未定義アドレスはどのノードに対しても設定されてはならないから」）と記載されています。

スコープには、スコープに対する**ゾーン**という考え方があります。スコープとゾーンはIPv6を構成する非常に重要な要素ですが、本書では、まずはユニキャストとエニーキャストにおいて存在しているリンクローカルスコープについて紹介したうえで、ゾーンについては3.6節で解説します。

リンクローカルスコープのIPv6アドレスは、対象となるネットワークインターフェースが同一リンク内でのみ一意なIPv6アドレスです。**グローバルスコープ**は、ネットワークインターフェースをインターネット全体の中で一意に識別できるIPv6アドレスです。

リンクローカルスコープのIPv6アドレスとしては、リンクローカルユニキャストアドレスが挙げられます。RFC 4007では、自分自身を示すループバックアドレスも、リンクローカルスコープに分類されると書かれています。ループバックアドレスは、想像上のリンクに接続された仮想的な**ループバックインターフェース**のものとして扱われています。

3.4 IPv6ノードに要求されるIPv6アドレス

IPv4では、1つのネットワークインターフェースに対して設定されるIPv4アドレスは基本的に1つだけです。これに対しIPv6では、リンクローカルユニキャストアドレスを要求するプロトコルがあるので、グローバルユニキャストアドレスが設定されているネットワークインターフェースであっても、同時にリンクローカルユニキャストアドレスの設定が必要になります。

IPv6ノードに要求されるIPv6アドレスについては、RFC 4291のSection 2.8でまとめられています。そこで示されているのは、ホストに要求されるIPv6アドレスと、ルータに要求されるIPv6アドレスです。

IPv6対応しているホストに設定が必要、もしくは対応が求められるIPv6アドレスには、以下のような種類があります。

- 各ネットワークインターフェースごとにリンクローカルユニキャストアドレス（3.5節参照）が必要

[†4] RFC 4007：S. Deering, B. Haberman, T. Jinmei, E. Nordmark, B. Zill, “IPv6 Scoped Address Architecture”, 2005年3月

CC BY NC SA

- ネットワークインターフェースに対するユニキャストまたはエニーキャストアドレス
- ループバックアドレス（`::1/128`）
- 全ノードアドレス（`ff02::1`）
- 各ユニキャストおよびエニーキャストアドレスに対応するSolicited-Nodeマルチキャストアドレス（11.3節参照）
- そのノードが参加する、その他マルチキャストアドレス（第11章参照）

IPv6対応しているルータでは、上記のIPv6アドレスに加えて、以下のIPv6アドレスにも対応が必要です。

- ルータとしての機能を提供しているネットワークインターフェースに対するサブネットルータエニーキャストアドレス（12.3節参照）
- ルータに対して設定されるその他のエニーキャストアドレス（第12章参照）
- 全ルータアドレス（`ff02::2`）

3.4.1 ルータ同士を/127で接続する際の注意

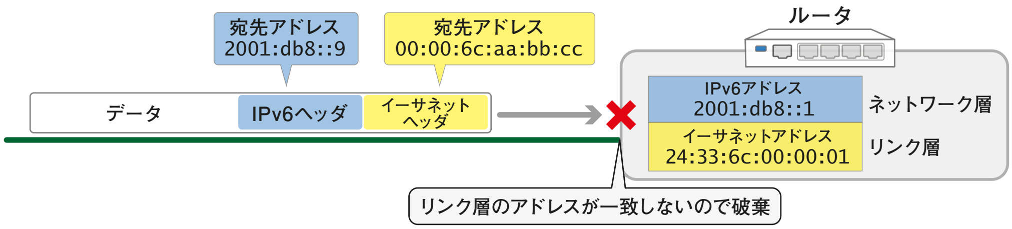
IPv6アドレスの長さは128ビットなので、サブネットプレフィックスを/127とすると、IPv6アドレスがネットワークで2つのみ存在できる状態になります。このような運用には、自分のIPアドレスがわかれば自動的に相手のIPアドレスもわかるという利便性があるほか、リンク層アドレスがないPoint-to-Pointなリンク上でのパケットのピンポン状態を防げるというメリットがあります。たとえば、PPPやIPv6 over IPv4トンネルなどではリンク層のアドレスがありません。そのため、受け取ったパケットが自分宛ではないかどうかをリンク層アドレスで判断できません。/127よりも短いプレフィックス長を利用すると、Point-to-Pointリンクの両端に接続された機器のIPアドレスではないIPアドレスを宛先とするパケットが、ピンポンのようにリンク上を行き来してしまう可能性があるのです。

イーサネットのようにリンク層アドレスがあれば、リンク層において自分宛ではないパケットを判別可能であり、そのようなパケットをネットワーク層へと渡す前に破棄できます。イーサネットでは、同一リンク上に存在する可能性がある他の機器宛のパケットを拾ってルータが転送してしまうことは、通常はありません。図3.1に、リンク層がイーサネットの場合の例を示します。ルータに対して他の機器宛のパケットが到着したとき、イーサネットヘッダに記載されている宛先イーサネットアドレスと、ルータのネットワークインターフェースのイーサネットアドレスが一致しない場合には、リンク層でパケットが破棄されネットワーク層へとパケットは到達しません。

これに対し、リンク層アドレスが存在しないPPPなどでは、ルータに到達した他の機器宛のパケットがそのままネットワーク層へと渡されることになります。そのため、/127よりも短いプレフィックス長で運用されているPoint-to-Pointリンクで接続している2つの機器が自分のIPv6アドレス宛でないパケットを受け取ってしまうと、それを相手に転送しようとしてしまいます。

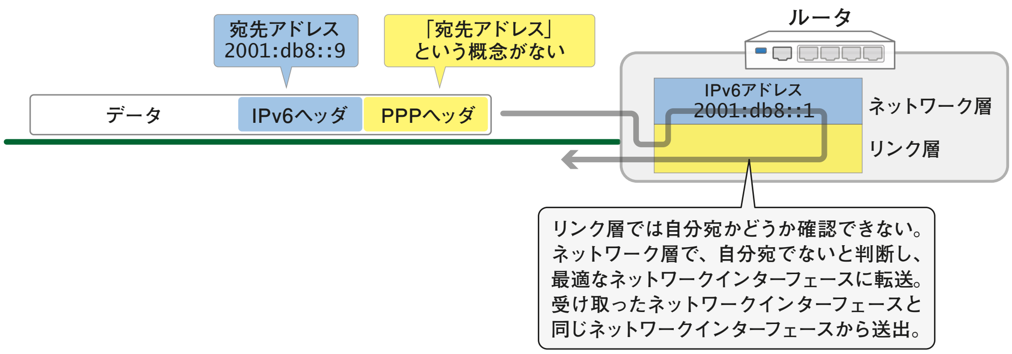
図3.2に、リンク層アドレスがないプロトコルで、宛先IPv6アドレスがルータのネットワー

CC BY NC SA



▶ 図 3.1　イーサネットの場合

クインターフェースとは異なり、かつ、同じネットワークプレフィックスに属する宛先 IPv6 アドレスに対するパケットを受け取った状況の例を示します。このとき、プレフィックス長は/64 だったとします。



▶ 図 3.2　リンク層にアドレスが存在しない場合

図 3.2 では、受信したデータが自分宛かどうかをリンク層では確認できないので、パケットがそのままネットワーク層へと渡されます。ネットワーク層では、そのパケットの宛先が自分ではないので、ルータとして転送を試みます。そのパケットの宛先 IPv6 アドレスは、パケットを受け取ったネットワークインターフェースが属するセグメントのものなので、パケットを受け取ったネットワークインターフェースへとパケットを転送します。

パケットの宛先が、リンクの両端に付いている IPv6 アドレスのいずれでもない場合、リンクの両端にあるルータがまったく同じ挙動を示す可能性があります。その結果、IPv6 ヘッダのホップリミットが 0 になるまでパケットが Point-to-Point リンク上を転送され続けてしまう「ピンポン状態」が発生してしまいます。

プレフィックス長が/127 であれば、このようなピンポン状態は発生しません。ピンポン状態が発生する条件として、IPv6 パケットの宛先アドレスがリンクの両端となっているネットワークインターフェースとは異なるものである必要がありますが、プレフィックス長が/127 であれば、そのリンク上で存在可能な IPv6 アドレスは 2 種類に限定されるからです。

この問題は、IPv6 に限定して発生する状況ではありません。IPv4 でも、環境が整えば同様の問題が発生します。しかし、IPv4 では IP アドレスを節約する必要性があることもあり、Point-to-Point なリンクでは/31 というサブネットを利用するのが一般的です。IPv4 アドレスの長さは 32 ビットなので、31 ビットをネットワーク部とし、最後の 1 ビットだけでホスト部

を示すわけです。こうすると同一セグメント内にIPアドレスが2つだけになるので、この問題が表面化しませんでした。

IPv4では/31がよく利用されている一方で、IPv6では、ルータ同士の接続で/127を利用するのは避けるべきであるというRFC 3627[†5]が2003年に発行されました。RFC 3627はInformationalなRFCであり標準ではありませんが、RFC化されていることもありそれなりに大きな影響力を持っていました。

RFC 3627において、ルータ同士の接続で/127を利用すべきでないとされている理由は、2つしかないIPv6アドレスを両方とも片方のルータが利用してしまう可能性があるからです。この状況が起きる背景には、第7章で説明するIPv6アドレス自動生成の仕組みと、サブネットに接続されたルータのネットワークインターフェースに設定されるサブネットルータエニーキャストアドレス（12.3節参照）の存在があります。ここでは簡単に問題が起こる状況を説明しておきます。

たとえば、/127のセグメントで片方のルータが通常のユニキャストIPv6アドレスを自動生成し、2つあるIPv6アドレスのうちの片方を利用することにしたとします。いま、もう片方のルータではIPv6アドレスが設定されていないとすると、最初のルータにおける重複アドレスの検知（7.6節で説明するDAD）は成功し、ネットワークインターフェースにそのアドレスが設定されます。このルータで、さらにサブネットルータエニーキャストアドレスを設定すると、2つあるIPv6アドレスのうちの両方が片方のルータに利用されることになります。この状況で、もう片方のルータにおいてIPv6アドレス自動生成を実行すると、DADに失敗することになります。結果として、そのルータでは、/127のセグメントのネットワークインターフェースにIPv6アドレスが付きません。

このような問題を避けつつ、ルータ同士が/127を利用できるようにするため、2011年にRFC 6164[†6]が発行されました。RFC 6164では、ルータ同士の接続において/127を利用する際にサブネットルータエニーキャストアドレスを利用することを禁止しています。また、/127を利用する際に後半64ビットがすべて0となるIPv6アドレスを利用することや、後半64ビットの上位128個の範囲（`ffff:ffff:ffff:ff7f`〜`ffff:ffff:ffff:ffff`）を利用することも禁止しています。

RFC 3627はInformationalなRFCでしたが、RFC 6164はStandards TrackのRFCです。RFC 6164のほうが重要視される分類ですが、それをより明確にするために、2012年にRFC 3627を廃止（Historic Status）するRFC 6547[†7]が発行されました。したがって、現在では、ルータ同士の接続において/127というプレフィックス長を使うことを妨げるRFCは存在しないことになります。

---

†5　RFC 3627：P. Savola, “Use of /127 Prefix Length Between Routers Considered Harmful”, 2003年9月

†6　RFC 6164：M. Kohno, B. Nitzan, R. Bush, Y. Matsuzaki, L. Colitti, T. Narten, “Using 127-Bit IPv6 Prefixes on Inter-Router Links”, 2011年4月

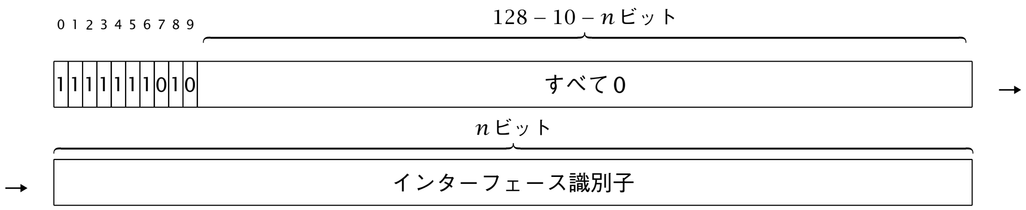
†7　RFC 6547：W. George, “RFC 3627 to Historic Status”, 2012年2月

3.5　リンクローカルユニキャストアドレス

リンクローカルユニキャストアドレスは、同一リンク内でのみ利用可能なリンクローカルスコープのIPv6アドレスです。実際、送信元もしくは宛先のIPv6アドレスとしてリンクローカルユニキャストアドレスが記載されたパケットは、ルータによる転送が禁止されています（RFC 4291のSection 2.5.6）。

単一のネットワークインターフェースには複数のIPv6アドレスが設定可能というIPv6の仕様のもと、RFC 4291のSection 2.8では、すべてのネットワークインターフェースにはリンクローカルユニキャストアドレスを設定する必要があるとしています。IPv6の基本的な機能にはリンクローカルユニキャストアドレスが自動的に設定されることを前提とした仕組みも少なくありません。第7章で説明するIPv6アドレスを自動設定する機能や、ルーティングプロトコルであるOSPFv3のプロトコル設計も、リンクローカルユニキャストアドレスの存在が前提となっています。リンクローカルユニキャストアドレスはIPv6を構成する重要な要素のひとつといえるでしょう。

RFC 4862[†8]のSection 5.3では、図3.3のような「先頭10ビットが1111 1110 10、それ以降のサブネットプレフィックスが0」のIPv6アドレス（`fe80::/10`）として、**リンクローカルアドレス**（リンクローカルスコープのIPv6アドレス）を生成するとあります。



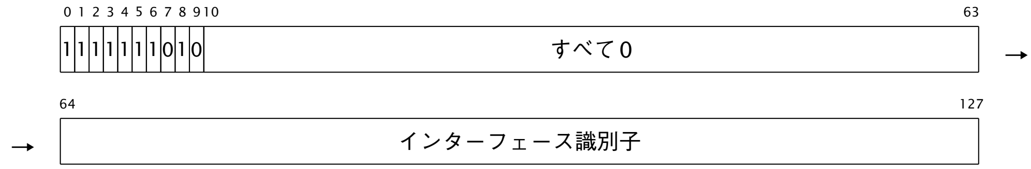
▶ 図3.3　リンクローカルアドレス

このときのインターフェース識別子の長さは、RFC 4862では任意とされています。したがって、インターフェース識別子の長さをnとすると、上位10ビットに続く$128 - 10 - n$ビットがすべて0です。

なお、RFC 4862が定義しているのはリンクローカルアドレスの生成方法であって、リンクローカルユニキャストアドレスそのものの定義ではありません。リンクローカルユニキャストアドレスそのものを定義しているRFC 4291のSection 2.5.6によると、リンクローカルアドレスのうち「サブネットプレフィックスが64ビット（つまり上位64ビットのうち後半54ビットがすべて0）で、下位64ビットがインターフェース識別子となるIPv6アドレス」（つまり`fe80::0/64`）を、リンクローカルユニキャストアドレスとしています。このため、リンクローカルユニキャストアドレスの場合は、RFC 4862においてインターフェース識別子の長さを示すnが64より大きな値になることはありません。

[†8] RFC 4862：S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration”, 2007年9月

CC BY NC SA



▶ 図3.4　リンクローカルユニキャストアドレス

NOTE

本書では、fe80::0/64のリンクローカルユニキャストアドレスを「リンクローカルアドレス」と表現することがあります。ff02::1などのリンクローカルスコープのマルチキャストアドレスに関しては、「リンクローカルマルチキャストアドレス」と表現します。

IPv4におけるリンクローカルアドレス

リンクローカルアドレスは、IPv6固有のものではありません。RFC 3927[†9]では、169.254.0.0/16がリンクローカルIPv4アドレスとして定義されています。

RFC 3927が発行されたのは2005年なので、IPv6が当初議論された時点では、IPv4用のリンクローカルアドレスは正式には存在していませんでした。169.254.0.0/16というアドレスは、主にWindowsにおいて、DHCPでIPv4アドレスを取得する前に独自にネットワークインターフェースに付けるIPv4アドレスとして知られていたものです。Windowsでは、DHCPのパケットを送出するときに送信元IPv4アドレスとして0.0.0.0を設定できないことから、このアドレスが利用されていました。これが後になって他のシステムでも利用されるようになり、2005年に正式なIPv4用のリンクローカルアドレスとしてRFC化されたのです。

3.6　スコープのゾーン

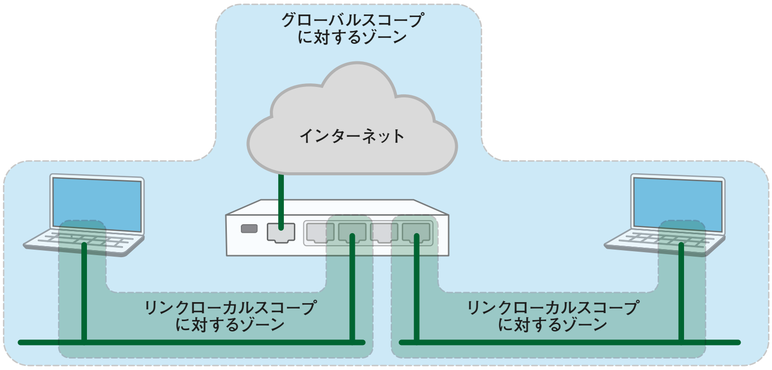
あるスコープを持つ特定のトポロジ的な範囲のことを、RFC 4007では**ゾーン**と呼んでいます。スコープがアドレス体系から決まるのに対し、ゾーンは個々のリンクを構成するインターフェースなどで決まります。どのようにしてゾーンが決まるかは、スコープごとに次のように整理できます。

- インターフェースローカルスコープ（11.2節）に対しては、ノード内の各インターフェースによって、1つのゾーンが形成される（マルチキャストの場合のみ）
- リンクローカルスコープに対しては、あるリンクとそのリンクに接続されたインターフェースによって、1つのゾーンが形成される（ユニキャスト、エニーキャスト、マルチキャストの場合）
- グローバルスコープに対しては、インターネットに接続されたすべてのリンクとネットワークインターフェースによって、1つのゾーンが形成される

†9　RFC 3927：S. Cheshire, B. Aboba, E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses”, 2005年5月

- インターフェースローカル、リンクローカル、グローバル以外のスコープについては、ネットワーク管理者によってゾーンが定義され、設定される必要がある

同じスコープに対応するゾーンがオーバーラップすることはありません。あるインターフェースが属するゾーンは、同じスコープに対するものであれば、必ず1つだけです。スコープが異なれば、複数のゾーンに同じインターフェースが属することはあります。たとえば、すべてのインターフェースはリンクローカルなゾーンに属すると同時にグローバルなゾーンにも属しています。実際、小さいスコープに対応するゾーンは、より大きなスコープに対応するゾーンに完全に内包されます。もちろん、その場合でも、小さいスコープに対応するゾーンがより大きなスコープに対応する複数のゾーンにまたがることはありません。図3.5に、スコープとゾーンの関係を模式的に示します。



▶ 図3.5　スコープとゾーンの関係

IPv6では、ゾーン内のパケットがゾーン外に転送されることが禁止されています。ゾーンが示す宛先に向かうパケットは、そのゾーンから抜け出すことができません。

NOTE

トンネルを利用した通信では、「パケットがトンネルを通じて遠隔に対して転送されている」ように思えるかもしれません。しかし、トンネルはゾーンを構成するので、ゾーン外に転送されるわけではありません。トンネルを通るパケットはゾーン内での転送になります。

ユニキャストとエニーキャストに必要なゾーンは、リンクローカルスコープに対応するものと、グローバルスコープに対応するものの、2つだけです。一方、マルチキャストでは多様なゾーンが必要になります。マルチキャストにおけるゾーンに関しては、11.4節で解説します。

NOTE

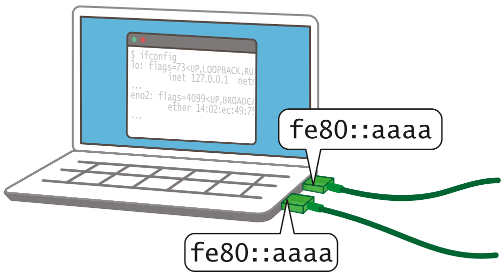
RFC 4007の5.Scope Zonesでは「インターフェースローカル、リンクローカル、グローバル以外のスコープについては、ネットワーク管理者によってゾーンが定義され、設定される必要がある」と書かれていますが、ユニキャストのサイトローカルスコープが廃止されたため、ユニキャストではネットワーク管理者がゾーンを定義するアドレスが存在せず、ネットワーク管理者がゾーンを定義するのは現在はマルチキャストのみになっています。マルチキャストにおいてゾーンを定義することは、ネットワークにおいてどこまでそのゾーンに属するマルチキャ

ストパケットが届くかを設定することでもあります。ただし、2021年現在、ルータを越えるマルチキャストの設定を行う機会があるネットワーク管理者は極端に少ないわけではありませんが、多いわけでもありません。そのため、RFC 4007で表現されているような、ネットワーク管理者がゾーンとなる範囲を定義し、設定する機会は、あまり多くなさそうです。

3.6.1 リンクローカルスコープにおけるゾーンの例

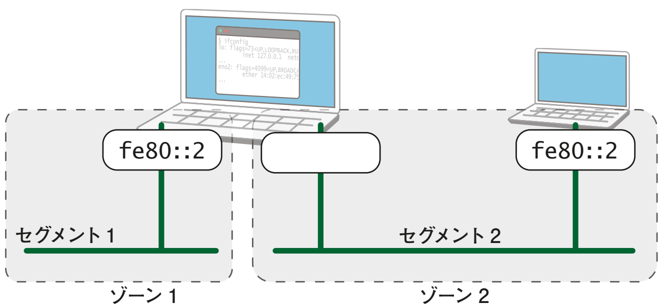
「ゾーンとゾーンの境界」はリンク上には存在せず、必ずノード内に存在します。そのため、ノード内の各インターフェースが同じIPv6アドレスを持ちながら、それぞれ異なるゾーンで利用されるという場合がありえます。

たとえば、1つのパソコンに2つのネットワークインターフェースが付いているとします。図3.6のように、両方のネットワークインターフェースに対して、まったく同じリンクローカルユニキャストアドレスを設定することも可能です。その2つのネットワークインターフェースは、それぞれ別のリンクローカルスコープのゾーンに属しているからです。



▶ 図3.6　1つのパソコン内で同じIPv6アドレス（リンクローカルユニキャストアドレス）

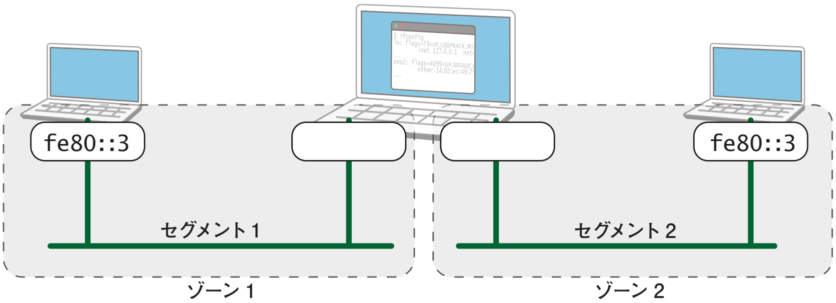
図3.7のように、同一リンクに接続している別のノードが同じIPv6アドレスを使っている状況も考えられます。この図の左側のゾーンで`fe80::2`というリンクローカルユニキャストアドレスが使われていて、右側のゾーンの別ノードで、同じリンクローカルユニキャストアドレスである`fe80::2`が使われても問題ありません。



▶ 図3.7　別ノードで同じリンクローカルユニキャストアドレス

このほか、図3.8のように、左側のゾーンの別ノードが`fe80::3`を使っていて、右側のゾーンの別ノードが同じ`fe80::3`を使っているというパターンも考えられます。

cc BY NC SA



▶ 図 3.8 隣のリンクで同じリンクローカルユニキャストアドレス

3.6.2 ゾーンインデックス

前項の例からわかるように、1つのノードに複数のネットワークインターフェースがある状態で、それぞれのゾーンに同じリンクローカルユニキャストアドレスが存在することも可能です。このため、「どのゾーンに属するリンクローカルユニキャストアドレスなのか」を指定しないとIPv6アドレスの一意性が確保できません。

IPv6では、あるインターフェースにおけるグローバルスコープ以外[†10]のアドレスがどのゾーンに属するものかをノード内で一意に識別するために、**ゾーンインデックス**（ゾーンID）という仕組みが使われます。ゾーンインデックスは、グローバルスコープではないIPv6アドレスを扱う状況において、システムの運用などに使われることがあります。

ゾーンインデックスのテキスト表記として、RFC 4007では以下のフォーマットを推奨しています。

- <address>%<zone_id>

ここで、「<address>」にはIPv6アドレス、「<zone_id>」にはゾーンごとに一意な文字列（ゾーン識別子）が入ります。たとえば、システム内で3番めのネットワークインターフェースに`fe80::1`というリンクローカルユニキャストアドレスが設定されていた場合、リンクに対応する番号を%記号の後に付記して`fe80::1%3`のように示すことができます。

<zone_id>で示されるゾーン識別子をどのように表現するかは実装により異なります。BSDの実装ではリンクとインターフェースが1対1だと仮定しており、結局のところ一意性も保証されることから、実装の都合上の簡易記法として「インターフェースの番号をリンクにおけるゾーン識別子として使う」例がよく見られます。ほかには、`eth0`、`le0`、`sl1ppp2`といったネットワークインターフェースの「名前」をゾーン識別子として利用する実装もあります。その場合、先の例は`fe80::1%eth0`のように表現されます。

†10 グローバルスコープの場合は、インターネット全体が1つのゾーンです。ゾーンが1つしかないので、どのゾーンに属するのかが自明であり、IPv6アドレスだけで一意性があることから、特にゾーンインデックスを指定する必要はありません。

cc BY NC SA

NOTE

RFC 4007では<zone_id>を「link index 2」のように表現しています。このような表現は「読みやすさのため」とされており、ノード内でゾーン識別子をどのように表現するかは実装によって異なります。

3.6.3 URIにゾーン識別子が含まれる場合の表記

「ゾーン識別子と組み合わせたIPv6アドレスのテキスト表現」をURIで使いたいことがあります。しかし、URIの構文（Generic Syntax）を定義しているRFC 3986では“This syntax does not support IPv6 scoped addressing zone identifiers”（「この構文はIPv6 scoped addressingのゾーン識別子はサポートしていない」）と明確に示してあり、ゾーン識別子とIPv6アドレスを組み合わせた表記をサポートしていません。これは、RFC 3986で定義されているURI構文では「%」という文字がエスケープ対象であり、IPv6アドレスのゾーン識別子を示すためにURI中に「%」をそのまま使うことがURI構文の仕様と競合するからです。

ゾーン識別子を含むIPv6アドレスをURIで使う場合の構文については、2013年にRFC 6874[†11]が発行されています。RFC 6874では、ゾーン識別子とIPv6アドレスの間にある%に対してPercent-Encodingを施した値である%25を利用することを定義しています。これにより、たとえばfe80::a%eth0を含むURLの表現は以下のようになります。

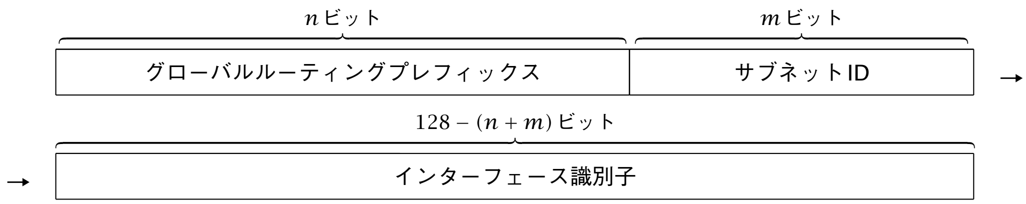
- `http://[fe80::a%25eth0]/`

NOTE

RFC 6874では、URI中では「%25」を使うことを必須としていますが、過去の実装では「%」を扱っていたものもあり、すべての実装が「%25」ではない可能性もあります。他のRFC同様に、RFCで標準仕様とされていることが必ずしもすべて仕様どおりに実装されているとは限らないという点に注意が必要です。

3.7 グローバルユニキャストアドレス

RFC 4291に記載されているIPv6のグローバルユニキャストアドレスは、図3.9のようなフォーマットになっています。



▶ 図3.9　グローバルユニキャストアドレス

nビット分のグローバルルーティングプレフィックスがあり、次にmビット分のサブネッ

†11 RFC 6874 : B. Carpenter, S. Cheshire, R. Hindenn, “Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers”, 2013年2月

トIDが続きます。nおよびmは可変長です。IPv6アドレスは128ビットなので、nとmを足した値を128から引いた値がインターフェース識別子となります。

RFC 4291のSection 2.5.1では、「最初の3ビットが000で開始するものを除くIPv6ユニキャストアドレスは、インターフェース識別子を64ビット長にする」ことが要求されています。そのため、たとえばグローバルルーティングプレフィックスが32ビットであった場合、各家庭へのサブネットIDとして残りの32ビットを利用し、家庭内での個別機器識別のためのインターフェース識別子用に残りの64ビットを利用するという運用が一般的です。

3.7.1 IPv6アドレスの/64境界について

IPv6の基本仕様には、IPv6アドレスの128ビットをサブネットプレフィックスとインターフェース識別子に分ける際の境界について明記されていません。しかし、IPv6アドレス体系の仕様であるRFC 4291では、「先頭3ビットが000で開始するものを除くすべてのユニキャストIPv6アドレスのインターフェース識別子は64ビット長である」とされています。ただし、IPv6アドレスにおいてサブネットプレフィックスの境界を/64としない状況を扱った唯一の例外的なRFCとして、RFC 6164における/127についての記述があります（3.4.1項参照）。

IPv6アドレスにおける/64境界に関しては、RFC 7421[†12]としてまとめられています。RFC 7421は、/64境界が広く採用されるようになった経緯や、関連するRFC、インターフェース識別子部分を可変長にした場合に発生し得る課題などをまとめたInformationalなRFCです。

RFC 7421によると、/64境界は、IPv6の初期デザインの時点で議論されていたとあります。その後、/80を境界とするという案があったものの、48ビット長のMACアドレスを置き換える形でEUI-64のMACアドレスの利用が普及するだろうと当時は予想されていたこともあり、/64境界でのインターフェース識別子がデファクトになりました。

IPv6の基本仕様は/64に限定されていないので、/64以外のプレフィックス長も利用は可能です。ただし、/64を強く意識しているプロトコルや、ある一定のプレフィックス長以上になると対応できないプロトコルなども存在するので、注意が必要です。

実装上の問題もあります。RFC 7421では、/80や/96などのプレフィックス長で動作できる転送機器が存在することが述べられています。しかし、プレフィックス長が64を超えると性能が劣化するルータが存在することも紹介されています。このような性能劣化はルータの設計によります。メモリや消費電力の有効利用、索引の速度上昇などを目指すために、64ビット境界を前提とした設計を採用するチップもあるからです。

まとめると、/64以外のサブネットプレフィックスも可能ではあるものの、デファクトとしては/64が採用されているのが現状であり、基本的には/64を利用しておくことが無難といえます。

[†12] RFC 7421：B. Carpenter, T. Chown, F. Gont, S. Jiang, A. Petrescu, A. Yourtchenko, “Analysis of the 64-bit Boundary in IPv6 Addressing”, 2015年1月

CC BY NC SA

3.8　ULA（Unique Local IPv6 Unicast Addresses）

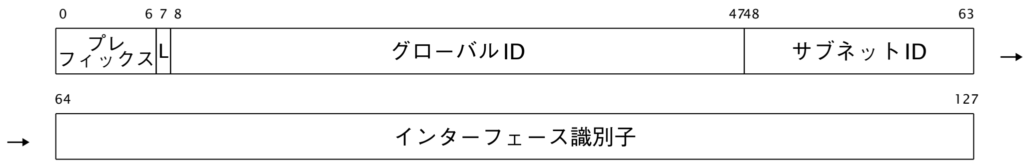
IPv6には、サイト内でのローカルな通信で利用するために、**ULA**（Unique Local IPv6 Unicast Address）というユニキャストIPv6アドレスが用意されています。ULAは、RFC 4193[†13]で定義されています。

RFC 4193ではULAの特徴として以下の6項目が示されています。

- グローバルにユニークなプレフィックス（高い確率での一意性）
- サイト境界でのフィルタリングを行いやすくするためのWell-knownなプレフィックス
- ULAによって運用されるサイトの統合や、ULA同士のプライベートな相互接続を、アドレスの競合や、ULAを利用しているインターフェースのリナンバリングを要求せずに行える
- サイト内での利用に際して、ISPに依存せず、インターネットへの接続を必要としない
- 事故によって経路やDNSでのリークが発生したとしても、他のサイトとの競合が発生しない
- アプリケーションがグローバルスコープのIPv6アドレスとして扱うことも可能

ULAは、サイト内でのローカルな通信だけでなく、限定された数のサイト間でのルーティングでの利用も想定されています。そのため、ULAのスコープ（3.3節）は、グローバルユニキャストアドレス同様にグローバルスコープです。ただし、グローバルなIPv6インターネットでULAの経路が広告されることは想定されていません（禁止されているわけでもないので注意してください）。なお、サイトローカルスコープというスコープもありますが、これはマルチキャスト用に規定されているスコープです（11.2節参照）。

ULAのフォーマットを図3.10に示します。



▶ 図3.10　ULAフォーマット

ULAの先頭7ビットはプレフィックスで、`fc00::/7`です。8ビットめは、ローカルであるかどうかを示すLビットで、1の場合がローカル、0の場合は未定義（「将来定義される」）となっています。このようにRFCに記載されたプロトコルには「将来定義される」とされたフィールドが多くあり、いつまでも定義されずにそのままになっていることも少なくありません。本書執筆時点でULAとして利用されるIPv6アドレスは、事実上は`fd00::/8`のみとなります。

IPv6アドレスにおけるサブネットプレフィックス（45ページの図2.7参照）にランダムな値が含まれることも、ULAの大きな特徴です。図3.10で「グローバルID」となっている40ビットが疑似乱数値となります。このフィールドでは、既知の数値などを利用して一意性を確保することは禁止されており、RFC 4086[†14]に基づく疑似乱数値を生成して利用することが求めら

†13　RFC 4193：R. Hinden, B. Haberman, “Unique Local IPv6 Unicast Addresses”, 2005年10月
†14　RFC 4086：D. Eastlake 3rd, J. Schiller, S. Crocker, “Randomness Requirements for Security”, 2005年6月

れています。この部分がランダムであることから、ULAに関連する経路、DNS、あるいはパケットが外に漏れたとしても、他のアドレスとの競合が発生しにくいとされています。

ランダムな40ビットのグローバルIDの後ろには、16ビットのサブネットIDフィールドが続きます。したがって、IPv6アドレスとしてのインターフェース識別子は64ビットとなっています。

サイトローカルアドレス

2000年代前半以前にIPv6を勉強された方は、**サイトローカルアドレス**と呼ばれるIPv6アドレスの存在を記憶されているかもしれません。サイトローカルアドレスは、2004年に、RFC 3879[†15]で廃止されました。

サイトローカルアドレスが廃止された主な理由は、「サイト」という概念で表される範囲が曖昧だったことでした。範囲が曖昧であったことから、運用管理者にとってデメリットが多く、アプリケーション開発者による実装でも利用しにくかったのです。

3.8.1 グローバルIDの乱数値の生成例

RFC 4193では、ULAのグローバルIDフィールドの値を、RFC 4086に準拠した疑似乱数値として生成することを求めています。そのうえで、グローバルIDのための疑似乱数値を生成するアルゴリズムの例が次のように紹介されています。

1. 現在時刻を64ビットのNTPフォーマットで取得
2. EUI-64識別子をシステムから取得。EUI-64識別子が存在しないシステムであれば、48ビットのMACアドレスからEUI-64を生成。もし、いずれも作成できない場合には、システムと関連があり適度な一意性がある識別子を利用する
3. 鍵を生成するために現在時刻とシステムに関連する一意性のある識別子を結合する
4. 鍵に対して、結果が160ビットとなるSHA-1ダイジェストを計算する
5. 160ビットのうちの下位40ビットをグローバルIDとして利用する
6. プレフィックス（`fc00::/7`）、Lビット（1）、上記で用意したグローバルIDの40ビットをつなぎ合わせ、ULAのIPv6アドレスプレフィックスとする

3.8.2 ULAとマルチアドレス

IPv4では、「イントラネット内部をプライベートIPv4アドレスで運用し、インターネットとの通信を行う場合にはIPv4 NATルータを経由する」という構成が使われることがあります。そのような既存環境をIPv6対応する場合に、ULAをプライベートIPv4アドレスのように使い、同じ構成のままで運用すればよいと思うかもしれません。

技術的には、IPv6でもNATを利用した構成をとることは可能です。IPv4のNATとは仕組みが違うものの、RFC 6296[†16]のNPTv6が利用できます（3.12.1項参照）。しかしIETFでは、

[†15] RFC 3879：C. Huitema, B. Carpenter, “Deprecating Site Local Addresses”, 2004年9月
[†16] RFC 6296：M. Wasserman, F. Baker, “IPv6-to-IPv6 Network Prefix Translation”, 2011年6月

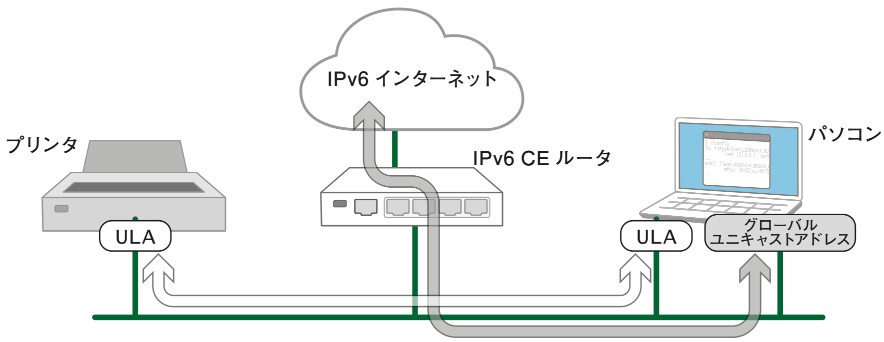
CC BY NC SA

IPv4と同じようなやり方でIPv6 NATを利用することは「特殊な場合を除き非推奨」としています。実際、2010年に発行されたRFC 5902[†17]には、「NATとファイアウォールを混同すべきではない」、「IPv6 NATは基本的に推奨しない」といった記述があります。

では、イントラネットなどの限定された範囲内での通信のためにULAを使っている環境で、イントラネット内部だけではなくIPv6インターネットとも通信したい場合、NATを使わずにどうすればいいのでしょうか。

IETFでは、そのような場合に「ULAとグローバルユニキャストアドレスによるマルチアドレス」を利用する方法を説明しているInformationalなRFC 7368[†18]を発行しています[†19]。IPv6は1つのネットワークインターフェースに複数のIPv6アドレスを設定することを前提とした設計になっているので、それを活用するというわけです。具体的には、ULAは限定された範囲のローカルな通信だけで利用し、インターネットとの通信にはグローバルユニキャストアドレスを使うことになります。

図3.11のような状況で説明しましょう。LAN内にある機器のうち、内部のみとの通信を行う機器にはULAのIPv6アドレスだけを設定します。この例ではプリンタがULAのみで運用されています。このプリンタにはグローバルユニキャストアドレスが設定されていないので、外部から直接プリンタと通信を行うことはできません。一方、パソコンについてはグローバルユニキャストアドレスとULAの両方のIPv6アドレスを設定しており、プリンタとの通信にはULAを、IPv6インターネットとの通信にはグローバルユニキャストアドレスを使うようにしています。



▶ 図3.11　ULAとグローバルユニキャストアドレスの併用

RFC 7368には以下のような記述があり、ここからもNPTv6に否定的なことが読み取れます。

> Note that unlike private IPv4 space as described in RFC 1918, the use of ULAs does not imply use of an IPv6 equivalent of a traditional IPv4 NAT [RFC3022] or of NPTv6

†17 RFC 5902：D. Thaler, L. Zhang, G. Lebovitz, “IAB Thoughts on IPv6 Network Address Translation”, 2010年7月

†18 RFC 7368：T. Chown,J. Arkko,A. Brandt,O. Troan,J. Weil, “IPv6 Home Networking Architecture Principles”, 2014年10月

†19 RFC 7368はホームネットワークに関するものですが、他のネットワークにおいてもULAに関する考え方は同様です。

CC BY NC SA

prefix-based NAT [RFC6296]. When an IPv6 node in a homenet has both a ULA and a globally unique IPv6 address, it should only use its ULA address internally and use its additional globally unique IPv6 address as a source address for external communications. This should be the natural behaviour given support for default address selection for IPv6 [RFC6724].
（参考訳）RFC 1918 におけるプライベート IPv4 空間とは異なり、ULA の利用では旧来の IPv4 NAT の IPv6 版や、NPTv6 によるプレフィックスに基づいた NAT の利用が意図されていない点に注意すること。homenet 内の IPv6 ノードが ULA とグローバルに一意な IPv6 アドレスを持つとき、ULA は内部での通信のみ使い、グローバルに一意な IPv6 アドレスは外部との通信のための送信元アドレスとして使うべきである。IPv6 では規定のアドレス選択ルールが用意されているので、これが自然な挙動となるべきである。

IPv6 CE ルータは ULA をどう扱うべきか

家庭や小規模なオフィスのネットワークの境界で IPv6 パケットの転送を担うノードのことを IPv6 CE ルータと呼びます。IPv6 CE ルータに関しては RFC 7084[†20] で要求仕様がまとめられていて、そこで IPv6 CE ルータが ULA をどう扱うべきかも示されています。たとえば、「ULA を利用したトラフィックが WAN 側のインターフェースから外に出ないようにすること」、「ULA に含まれるランダムな部分を生成したルータが再起動しても ULA プレフィックスが記憶されること」などが要求されています。

RFC 7084 では、IPv6 CE ルータが扱っている IPv6 プレフィックスが ULA のみの場合、自分自身をデフォルトルータとして広告することを禁止しています。具体的には、Router Advertisement メッセージの Router Lifetime フィールド（6.2.1 項参照）として、0 よりも大きい値を設定することを禁止しています。つまり、「WAN 側が IPv6 インターネットとの接続がない一方で、LAN 側で ULA のアドレスが運用されている場合には、自分自身をデフォルトルータとして広告するな」ということになります。

ただし、RFC 7084 には IPv6 NAT である NPTv6 を禁止していると読める文言は見当たりません。そのため、「WAN 側に IPv6 インターネットとの接続性がある状態で、LAN 側では ULA のみを広告しつつ、デフォルトルータとして自身を広告する」ようなことは禁止されていないと思われます。

3.9 IPv4-Mapped IPv6 アドレス

IPv6 アプリケーションが、IPv4 で容易に通信するためのアドレスとして、**IPv4-Mapped IPv6 アドレス**（IPv4 射影 IPv6 Address）があります。IPv4 との互換性のために標準化されている IPv4-Mapped IPv6 アドレスは、IPv4 アドレスを表現する手段としても利用されます。たとえば、デフォルト IPv6 アドレス選択のためのポリシーテーブルでは、IPv4-Mapped IPv6 アドレスによって IPv4 を表現します（13.1.3 項参照）。

---

†20 RFC 7084 : H. Singh,W. Beebee,C. Donley,B. Stark, “Basic Requirements for IPv6 Customer Edge Routers”, 2013 年 11 月

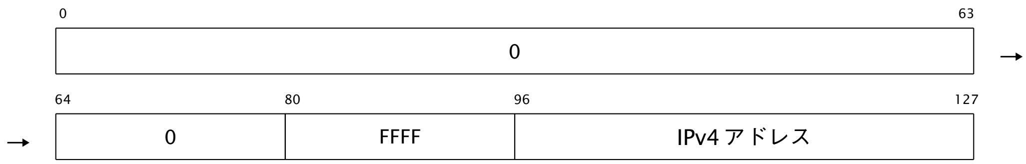
CC BY NC SA

IPv4-Mapped IPv6アドレスは、IPv6パケットの送信元や宛先アドレスとして利用するためのアドレスではないので、意図せず使ってしまわないように注意が必要です。

NOTE

紛らわしいかもしれませんが、RFC 4659[†21]やRFC 4798[†22]では、BGP-MPLS利用時に、BGP Next Hopフィールドとして IPv4-Mapped IPv6アドレスが活用される場合もあると紹介されています。
また、現在は廃止されているものの、RFC 2765[†23]にて規定されていたSIIT（Stateless IP/ICMP Translation Algorithm）の仕様では、IPv4-Mapped IPv6アドレスを設定したIPv6パケットの送受信が前提とされていました。

IPv4-Mapped IPv6アドレスでは、IPv6アドレスの上位80ビットをすべて0、それに続く16ビットを1、最後の32ビットをIPv4アドレスとします。したがって、IPv6アドレスとしては::ffff:0:0/96というフォーマットをしています。図3.12にIPv4-Mapped IPv6アドレスのフォーマットを示します。



▶ 図3.12　IPv4-Mapped IPv6アドレス

IPv4-Mapped IPv6アドレスは、::ffff:[IPv4アドレス]のように表記されることもあります。たとえば、192.0.2.201というIPv4アドレスの相手と通信するときに使うIPv4-Mapped IPv6アドレスであれば、::ffff:192.0.2.201のように表現されます。

3.9.1　ノード内でのIPv4-Mapped IPv6アドレスの扱い

IPv6のための基本的なソケット拡張APIを記したRFC 3493[†24]と、アプリケーションがIPv6対応するときに考慮するべき内容をまとめたRFC 4038[†25]では、ノード内部でのIPv4-Mapped IPv6アドレスの扱いが説明されています。IPv6アプリケーションでは、IPv4-Mapped IPv6アドレスを利用することで、図3.13のようにIPv4からとIPv6からの両方の通信を単一のソケットで扱えるようになります。

IPv6アプリケーションがパケットを送信する場合には、IPv4-Mapped IPv6アドレスを指定することで、IPv4アドレス宛に送信できます。逆に、IPv4から受け取るパケットの送信元

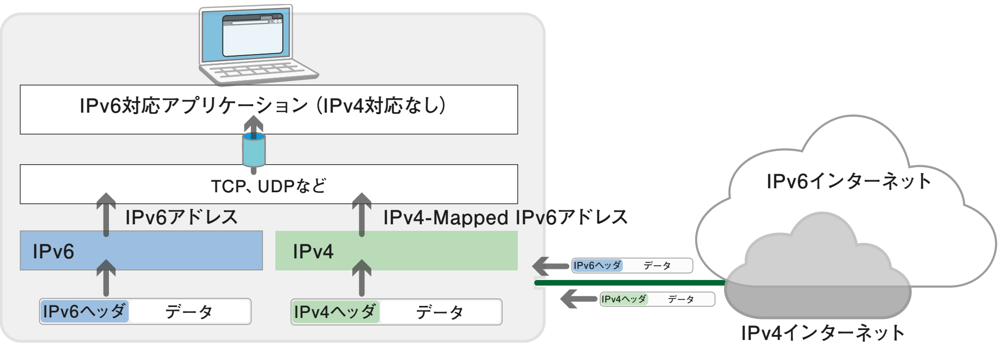
†21　RFC 4659：J. De Clercq, D. Ooms, M. Carugi, F. Le Faucheur, “BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN”, 2006年9月

†22　RFC 4798：J. De Clercq, D. Ooms, S. Prevost, F. Le Faucheur, “Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)”, 2007年2月

†23　RFC 2765：E. Nordmark, “Stateless IP/ICMP Translation Algorithm (SIIT)”, 2000年2月

†24　RFC 3493：R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens, “Basic Socket Interface Extensions for IPv6”, 2003年2月

†25　RFC 4038：M-K. Shin, Y-G. Hong, J. Hagino, P. Savola, E. M. Castro, “Application Aspects of IPv6 Transition”, 2005年3月



▶ 図 3.13　単一のソケットで IPv4-Mapped IPv6 アドレスを利用する場合

や宛先は、IPv4-Mapped IPv6 アドレスによって表現できます。アプリケーション内部で利用する IP アドレスを IPv6 のみに限定する一方で、機器から送出されるパケットについては、IPv4-Mapped IPv6 アドレスを使うことで IPv4 パケットとして送出できるというわけです。

IPv6 ソケットで IPv4 パケットも受け取れるので、IPv6 のみの通信を行っているつもりでも、実は IPv4 での通信を行っている場合もあります。たとえば RFC 3493 では、IPv4-Mapped IPv6 アドレスを、IPv6 ソケットの初期値として利用することが許可されています。明示的に指定しない限りは、IPv6 だけで通信をしているつもりが IPv4 での通信も受け付けるようになっている場合があるのです。しかも、RFC 3493 の Section 5.3 によれば、`IPV6_ONLY` のソケットオプションを利用して明示的に IPv6 のみを受け取れるようにしたとしても、IPv6 パケットとして受け取る IPv4-Mapped IPv6 アドレスでの通信は受け付けてしまいます。そのため、IPv6 対応のアプリケーションでは、IPv4-Mapped IPv6 アドレスの扱いに注意が必要です。

3.9.2 IPv4-Mapped IPv6 アドレスが抱えるセキュリティ上の懸念

IPv4-Mapped IPv6 アドレスが抱えるセキュリティ上の懸念が、RFC 4942[†26] の Section 2.2 で紹介されています。RFC 4942 では、当初はノード内での利用のみが想定されていた IPv4-Mapped IPv6 アドレスが RFC 2765 の SIIT によってネットワークに送出されるパケットで利用される状況になったことにより、以下の 3 種類の攻撃が可能になったとしています。

- `::ffff:127.0.0.1` という送信元 IPv6 アドレスのパケットを受信させることで、アプリケーションに対してローカルホストからのパケットであると誤認させ、アクセス制限を回避できる可能性がある
- IPv4-Mapped IPv6 アドレスを宛先とする IPv6 パケットを送信することで、IPv4 でのパケットフィルタルールを回避できる可能性がある
- IPv4-Mapped IPv6 アドレスを IPv4 アドレスへと変換したうえで、IPv4 パケットを生成するようなプロトコルを踏み台とした攻撃が考えられる

[†26] RFC 4942 : E. Davies, S. Krishnan, P. Savola, “IPv6 Transition/Co-existence Security Considerations”, 2007 年 9 月

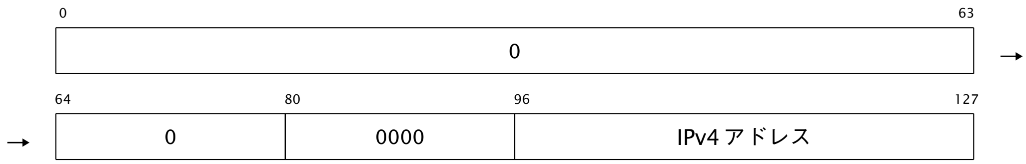
cc BY NC SA

このようなセキュリティ上の問題があることから、IPv4-Mapped IPv6アドレスが利用できない環境もあります。たとえば、OpenBSDにはセキュリティ上の理由でIPv4-Mapped IPv6アドレスが実装されていません。

3.10　IPv4-Compatible IPv6アドレス

かつては、IPv4-Compatible IPv6アドレス（IPv4互換IPv6アドレス）と呼ばれる`::/96`のIPv6アドレスが定義されていましたが、RFC 4291によって廃止されました。廃止の理由は、このIPv4-Compatible IPv6アドレスを使う仕組みが作られなかったためです。

IPv4-Compatible IPv6アドレスも、IPv4-Mapped IPv6アドレス同様に、`::[IPv4アドレス]`のように表記されることもあります。



▶図3.14　IPv4-Compatible IPv6アドレス

3.11　例示用IPv6アドレス

本書では、IPv4アドレス、IPv6アドレス、AS番号など、いくつかの「値」を例示していますが、そうした値はできるだけ実在しないものを選んでいます。インターネット上の通信で利用できる「値」は限られた範囲内のものなので、下手な「値」を使ってしまうと、実際に使われている場合に問題が発生する可能性があるからです。

例示のための値が、実在する値と競合するのを避けるために、サンプルとして使う値には一定の作法があります。特殊な用途のIPアドレスについてまとめたRFC 6890によると、ドキュメンテーションで使うために予約されているIPv6アドレスの値は、RFC 3849[†27]で下記のように定められています。

- `2001:db8::/32`

なお、RFC 5737[†28]で定義されている例示用IPv4アドレスは、以下の3ブロックです。

- `192.0.2.0/24`：TEST-NET-1
- `198.51.100.0/24`：TEST-NET-2
- `203.0.113.0/24`：TEST-NET-3

RFC 7042[†29]では、例示用MACアドレスが以下の範囲として定義されています。

---

†27　RFC 3849：G. Huston, A. Lord, P. Smith, "IPv6 Address Prefix Reserved for Documentation", 2004年7月

†28　RFC 5737：J. Arkko, M. Cotton, L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", 2010年1月

†29　RFC 7042：D. Eastlake 3rd, J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", 2013年10月

- ユニキャスト ： 00-00-5E-00-53-00 〜 00-00-5E-00-53-FF
- マルチキャスト ： 01-00-5E-90-10-00 〜 01-00-5E-90-10-FF

本書でも、これらのRFCに従ったものを例示では主に利用しています[†30]。皆さんも何らかのIPアドレスなどを例示するときは、これらの値を利用することをお勧めします。

例示用の値に関するRFCを下記にまとめます。

- RFC 2606[†31] : Reserved Top Level DNS Names
- RFC 3849 : IPv6 Address Prefix Reserved for Documentation
- RFC 5398[†32] : Autonomous System (AS) Number Reservation for Documentation Use
- RFC 5737 : IPv4 Address Blocks Reserved for Documentation
- RFC 6761[†33] : Special-Use Domain Names
- RFC 6890 : Special-Purpose IP Address Registries[†34]
- RFC 7042 : IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters

3.12 ユーザへのIPv6アドレス割り当て

IPv4では、「ISPからユーザに割り当てられるIPv4アドレスが1つ」というケースが一般的でした。IPv6では、ユーザへのIPv6アドレス割り当てとして、IPアドレスではなくネットワークプレフィックスを利用するのが一般的です。これは、IPv6アドレス空間がIPv4アドレス空間と比べて広大であるためです。

ISPから、どれぐらいの大きさのIPv6ネットワークプレフィックスをユーザに割り当てるのかに関しては、さまざまな議論があります。

IAB（Internet Architecture Board）によって2001年に発行されたRFC 3177[†35]では、一般的には/48、必要なサブネットが1つだけであるとわかっている場合は/64、接続デバイスが1つだけであると明確にわかっている場合には/128を割り当てるとありました。

しかし、このRFC 3177は、2011年に発行されたRFC 6177[†36]によって上書きされています。RFC 6177は、RFC 3177が推奨する割り当てサイズのデフォルト値を廃止し、割り当てサ

†30 ただしBGPの解説を行うには、例示用IPv4アドレスブロックが/24と小さすぎる場合があったり、例示用AS番号が極度にわかりにくかったりすることから、RFCに従わない場合もあります。また、MACアドレスが複数必要となる場合にも、IANAに割り当てられたEUI-48の番号以外を使うこともあります。155ページの「NOTE」で触れているように、IANAに返却されたアドレスを例示用に利用する場合もあります。

†31 RFC 2606：D. Eastlake 3rd, A. Panitz, “Reserved Top Level DNS Names”, 1999年6月

†32 RFC 5398：G. Huston, “Autonomous System (AS) Number Reservation for Documentation Use”, 2008年12月

†33 RFC 6761：S. Cheshire, M. Krochmal, “Special-Use Domain Names”, 2013年2月

†34 RFC 5156（M. Blanchet, “Special-Use IPv6 Addresses”, 2008年4月）とRFC 5735（M. Cotton, L. Vegoda, “Special Use IPv4 Addresses”, 2010年1月）は、RFC 6890によって廃止されています。

†35 RFC 3177：IAB, IESG, “IAB/IESG Recommendations on IPv6 Address Allocations to Sites”, 2001年9月

†36 RFC 6177：T. Narten, G. Huston, L. Roberts, “IPv6 Address Assignment to End Sites”, 2011年3月

CC BY NC SA

イズは運用コミュニティの課題である[†37]としました。そのため、現在ではユーザへの一般的な割り当てサイズという概念はなくなりました。

3.12.1 IPv6でのNATに関する議論

IPv4では、ISPから各家庭に割り当てられるIPv4アドレスが1つだけなので、NATを利用して複数機器がインターネットに接続できるようにする必要がありました。IPv6で単一のIPアドレスではなくネットワークプレフィックスが割り当てされるようになれば、そうした当初のNATの必要性は消えます。

3.8.2項でも触れたように、IPv6 NATに対するIABの考え方はRFC 5902で示されています。そこでは、サイトマルチホーミングなどにおけるIPv6 NATの有用性を認めつつも、インターネットにおけるEnd-to-Endでの接続性を維持するためにIPv6 NATを推奨しないことが明記されています。一方、IPv6でのマルチホーム環境やマルチプレフィックス環境の運用方法（13.2節参照）などをまとめたRFC 7157[†38]では、IPv6におけるNATを中間期の解決策となりうるとしています。

実際、IPv6においてNATがまったく存在しないわけではありません。IPv6環境同士を結ぶNATはNAT66と呼ばれます。実験的な位置づけのRFCですが、RFC 6296ではステートレスにNAT66を行うNPTv6（IPv6-to-IPv6 Network Prefix Translation）と呼ばれる仕組みが規定されています。日本国内では、NTTのフレッツ光ネクストにおいて、IPv6インターネットへの接続性のためにIPv6 PPPoE（A.3節参照）でNAT66が利用されています。

NOTE

NAT66については、IPv4 NATの場合と異なり、「LAN側で利用するためのIPv6アドレスをどうするのか」を考慮する必要があります。そもそもIPv6には、プライベートIPv4アドレスと同じ仕組みが存在していないからです。IPv6のULA（3.8節参照）がプライベートIPv4アドレスと同等のものであると勘違いされがちですが、ULAはグローバルスコープのIPv6アドレスであり、プライベートIPv4アドレスとはまったく違うものです。

IPv6におけるNATの必要性がセキュリティという観点で語られることもあります。NAT機器を導入すると、内部から外部への通信に応じて動的なフィルタが追加されます。この挙動はSFI（Stateful Filter Implementation）と呼ばれ、これにより接続性を確立する前に外部から内部へのパケットが転送できなくなります。この性質によって「外部からの侵入難易度が若干高くなる」ことから、「NATが使われないIPv6はIPv4よりも攻撃を受けやすい」と考えられることさえあります。

このような考え方に対し、RFC 4864[†39]では、NAT機器で実現される簡易なセキュリティ機能は「ないよりマシ程度」（‘better than nothing’ level of protection）であると反論してい

†37 RFC 6177のAbstractには、次のように書かれています。“The exact choice of how much address space to assign end sites is an issue for the operational community.”

†38 RFC 7157：O. Troan, D. Miles, S. Matsushima, T. Okimoto, D. Wing, “IPv6 Multihoming without Network Address Translation”, 2014年3月

†39 RFC 4864：G. Van de Velde, T. Hain, R. Droms, B. Carpenter, E. Klein, “Local Network Protection for IPv6”, 2007年5月

CC BY NC SA

ます。そして、NATを利用せずにNATと同等のセキュリティレベルを確保する方法として、LNP（Local Network Protection）と呼ばれるものが紹介されています。これは、SFIをファイアウォールで実装することで、「IPv4 NATがセキュリティ上貢献していると信じられている機能」をIPv6においても実現するものだといえます。

このように、IPv6においてもNATは必要であるという主張がある一方で、IETFをはじめとしてIPv6におけるNATの利用には反対する声が大きいのが現状です。実際、IETF界隈では、NTTフレッツ網におけるNAT66に否定的な意見も見かけます。

IPサービスにおける前提は時代によって変わる

IABによるRFC 6250[†40]では、IPサービスにおける前提が時間とともにどのように変化したのかを述べるために、「よくある勘違い」がいろいろ紹介されています。Informationalな RFCであり、読み物のような扱いではありますが、通信の対称性を期待してはならない話や、エラーメッセージが必ずしも受け取れない場合もあること、アドレスが時間とともに変化することがあること、ブロードキャストやマルチキャストがリンク内で使えない場合があることなど、NATについて考えるときにも考慮が必要なさまざまな注意点が記載されています。

3.12.2 プライバシーの問題

IPv4ではIPアドレスの数が限られていたので、複数ユーザ間でIPv4アドレスを使い回す運用が行われていました。ISPを利用してインターネットに接続するユーザにとっては、接続し直すたびにIPv4アドレスが変化する環境が一般的だったといえます。

IPv6では、ユーザがISPへの接続を切断してから再接続しても、前の接続時と同じIPv6ネットワークプレフィックスが割り当てられる場合が多くありえます。毎回同じIPv6ネットワークプレフィックスが利用されることで、事実上匿名性が薄れ、プライバシー情報を収集されやすくなる可能性があります。

IPv6ネットワークプレフィックスが毎回変化したとしても、下位64ビットの部分が同一になることで、個人を特定可能になる場合もあるでしょう。特にIPv6アドレスについては、第7章で後述するIPv6アドレスの自動設定（SLAAC）を利用することから、IPアドレス自身が個人を特定し得る識別子となります。SLAACで、ネットワークインターフェースに設定された世界で一意のIEEE識別子からIPv6アドレスの下位64ビット部分を生成する方法を使う場合、ネットワークを移動するなどしてネットワークのプレフィックスが変わっても、ユーザの識別および追跡が可能となってしまうからです。IPアドレスはインターネットにおける通信の根本であり、通信中に隠すことが難しいので、IPアドレスが個人を特定し得る識別子となることにはプライバシー情報漏洩の危険が伴います。

IPアドレスそのものを利用した追跡は、IPv6でSLAACを利用することにより発生する問題

[†40] RFC 6250：D. Thaler, “Evolution of the IP Model”, 2011年5月

cc BY NC SA

です[†41]。そのため、SLAACを使わずにDHCPv6を利用すれば、この問題によるプライバシー情報漏洩は回避可能です。また、SLAACを利用しつつプライバシーを確保する手法として、SLAACによるIPv6アドレス自動生成に対するプライバシー拡張がRFC 8981[†42]で定義されています[†43]。

RFC 8981では、通常のSLAACを拡張したTemporary IPv6アドレス（匿名IPv6アドレス）が定義されています（7.5節参照）。これは、疑似乱数値を持ち、時間とともに変化するIPv6アドレスです。Temporary IPv6アドレスの値は定期的に変更することが推奨されていて、上位層のアプリケーションなどが利用していない古いTemporary IPv6アドレスは新しいTemporary IPv6アドレス生成後に削除することも可能です。

†41　HTTPメッセージを解析するなどの手法による追跡はIPv4でも可能です。

†42　RFC 8981：F. Gont, S. Krishnan, T. Narten, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6”, 2021年2月

†43　RFC 8981はRFC 4941を、RFC 4941はRFC 3041（T. Narten, R. Draves, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”, 2001年1月）をそれぞれ上書き廃止する形で、改訂したものです。

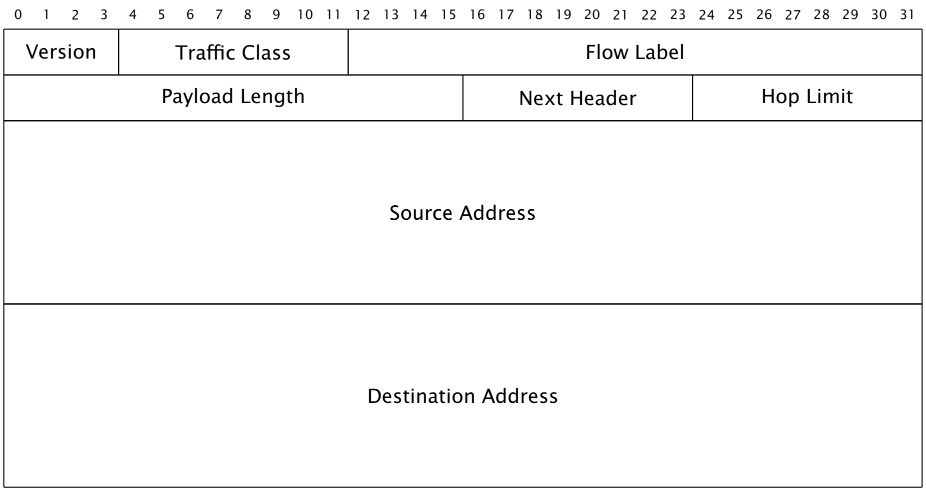
第4章

IPv6パケットの構成

さまざまなルータが相互接続をするためには、IPv6パケットがどのようなフォーマットで送受信されるのかについての取り決めが重要になります。相互接続を実現するために、IPv6パケットの先頭に付いているヘッダのフォーマットに関して世界中のルータの共通認識が必要なのです。IPv6の基本仕様を示しているRFC 8200[†1]でも、IPv6ヘッダのフォーマットが主要な内容になっています。IPv6の挙動を知るには、IPv6ヘッダのフォーマットと、各フィールドの意味を知る必要があります。

4.1 IPv6ヘッダの各フィールド

では、実際のIPv6ヘッダフォーマットを見てみましょう。IPv6ヘッダのフォーマットはRFC 8200で図4.1のように定義されています。



▶ 図4.1　IPv6ヘッダフォーマット

†1　RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017年7月

IPv6ヘッダの各フィールドは以下のような意味を持ちます。

- Version（4ビット）

　インターネットプロトコルのバージョン番号を示しており、「6」を示す0110になります。IPv4では、この部分が「4」を示す0100でした。

- Traffic Class（8ビット）

　上位6ビットがDiffservで利用するDSCP（Differentiated Services Code Point）を格納するDS Fieldで利用され（RFC 2474[†2]）、下位2ビットがECN（Explicit Congestion Notification）で利用されます（RFC 3168[†3]）。なお、RFC 2474には「IPv4のToSおよびIPv6のTraffic Classという名称をDS fieldへと変更する」とありますが、これはRFC 3260[†4]で「DS fieldはTraffic Classの上位6ビットとする」と訂正されています。

- Flow Label（20ビット）

　通信のフローを識別するのに使うフローラベルを設定するフィールドです。フローラベルについては101ページで説明します。

- Payload Length（16ビット）

　IPv6パケットのヘッダ部分に続くペイロードの長さを示します。ペイロードというのは、ヘッダに続く部分のことです（1.1.1項参照）。

　ルータは、受け取ったパケットを処理する際、このPayload Lengthフィールドで「そのパケットの長さがどれだけあるのか」を知ることができます。IPv6ヘッダは40オクテットの固定長なので、Payload Lengthフィールドの値にIPv6ヘッダの長さである40を足した値が、IPv6パケット全体の長さになります。

　Payload Lengthフィールドの値にはIPv6ヘッダの長さは含まれません。ただし、IPv6ヘッダに続くIPv6拡張ヘッダが存在する場合、それらのIPv6拡張ヘッダの長さはPayload Lengthに含まれます。

- Next Header（8ビット）

　IPv6ヘッダの直後に続くヘッダを示します。このフィールドで利用されるプロトコル番号はIANAによって割り当てられており、IPv4のProtocol Typeフィールドで利用される値と同一です。たとえば、次がTCPであれば6、UDPであれば17、ICMPv6であれば58となります（RFC 5237[†5]）。

　IPv6拡張ヘッダがIPv6ヘッダの直後に続く場合もあります。IPv6拡張ヘッダに関しては4.3節で説明します。

---

†2　RFC 2474：K. Nichols, S. Blake, F. Baker, D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, 1998年12月

†3　RFC 3168：K. Ramakrishnan, S. Floyd, D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP”, 2001年9月

†4　RFC 3260：D. Grossman, “New Terminology and Clarifications for Diffserv”, 2002年4月

†5　RFC 5237：J. Arkko, S. Bradner, “IANA Allocation Guidelines for the Protocol Field”, 2008年2月

cc BY NC SA

- Hop Limit（8ビット）

  IPv6パケットが転送される上限を示します。IPv6パケットが転送されるたびに1減算され、0になるとIPv6パケットが破棄されます。

  Hop Limitフィールドの値は、IPv6パケットを送信する機器が設定します。そして、そのIPv6パケットがルータによって転送されるたびに1ずつ減らされます。Hop Limitフィールドの値が0になったIPv6パケットは破棄され、それによってIPv6パケットが永遠にインターネット上を転送され続けながら彷徨い続けることが回避されます。Hop Limitフィールドのような仕組みが存在しない場合、たとえば、経路上にループが発生して、同じ経路上をぐるぐると転送され続けるパケットが存在したとしても、各ルータは単純に受け取ったパケットを転送し続けるだけなので、パケットが永遠に破棄されない状態ができてしまうおそれがあります。Hop Limitフィールドは、そのような状況が発生しないようにするための仕組みであり、IPv6パケットにとっての安全装置のようなものです。

- Source Address（128ビット）

  IPv6パケットが生成された送信元を示します。

- Destination Address（128ビット）

  IPv6パケットの宛先を示します。

Source AddressフィールドとDestination Addressフィールドは、128ビットのIPv6アドレスを指定するのに使うので、いずれも128ビットです。2つのフィールドを合計すると32オクテットであり、サイズだけに着目すると「IPv6ヘッダの大部分はIPv6アドレスを示すフィールド」だといえます。

Payload LengthフィールドとIPv6 Jumbogram

Payload Lengthフィールドの長さが16ビットなので、IPv6におけるペイロード長の最大値は、16ビットで示せる最大の長さの65,535オクテットになります。しかし、IPv6にはJumbogramという仕組みがあり、4.3節で後述するIPv6拡張ヘッダのHop-by-Hopオプションを利用することで、65,536オクテット以上のペイロードを持つIPパケットも利用可能です。IPv6 JumbogramはRFC 2675[†6]で定義されています。

IPv6 Jumbogramを利用するには、ノードが接続しているリンクで、一度に巨大なネットワーク層のデータを転送できる必要があります。具体的には、リンクで送信できるネットワーク層のデータの最大値のことをMTU（Maximum Transmission Unit）と呼ぶので、MTUが65,575オクテットよりも大きなリンクが必要です。インターネットでリンク層として採用されていることが多いイーサネットのMTUが通常は1500オクテットであることを考えると、IPv6 Jumbogramが広範なインターネットで使われることはあまりなさそうです。

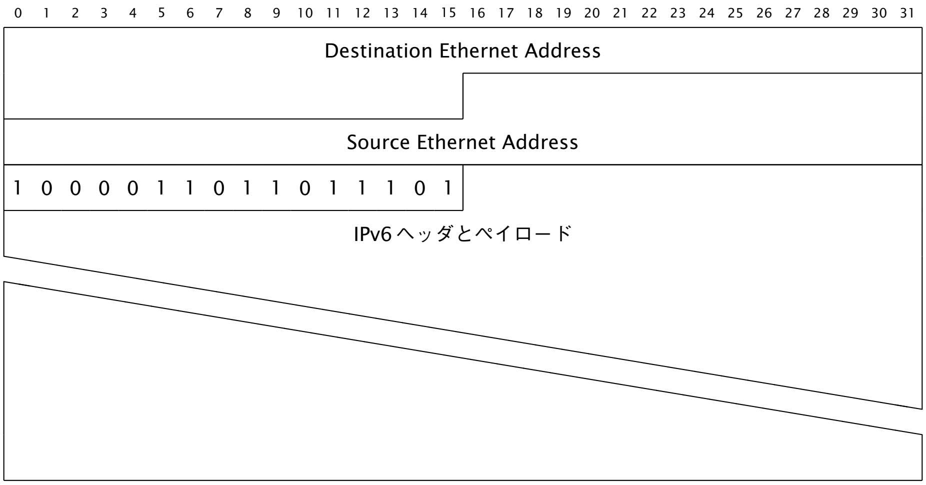
さらに、UDPヘッダには16ビット長のLengthフィールドがあることや、TCPのMSSオプションと緊急フィールドが16ビットに制限されていることを考えると、IPv6 Jumbogramを実際のネットワークで利用可能にするためにはTCPやUDPへの変更も必要です。

cc BY NC SA

IPv6ノードに要求される機能（IPv6 Node Requirements）をまとめたInformationalなRFCでは、2006年に発行されたRFC 4294[†7]（Obsolete）とRFC 6434[†8]（Obsolete）ではオプショナルな機能として紹介されていました。しかし、RFC 6434を上書きするRFCとして2019年に発行されたRFC 8504[†9]では、「デプロイされていない」という理由でJumbogramに関する記述が丸ごと削除されています。

4.1.1　イーサネットのEthernet Typeフィールド

イーサネットにおけるIPv6パケットの扱いはRFC 2464[†10]で規定されています。IPv6パケットは、イーサネット上では図4.2のようにカプセル化されます。



▶図4.2　IPv6パケットを含むイーサネットフレーム

1000011011011101の部分は、イーサネットフレームのEthernet Typeフィールドです。IPv6パケットを運ぶイーサネットフレームの場合、Ethernet Typeは0x86ddなので、1000011011011101となります。

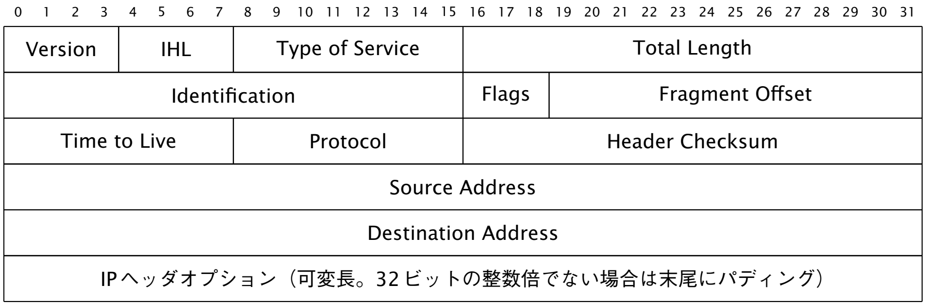
4.1.2　IPv4ヘッダとの違い

IPv6ヘッダのフォーマットは、IPv4ヘッダのフォーマットから大幅に変更されています。どんな点が変更されているかを知ることで、IPv6の設計思想の一部を垣間見ることができます。IPv6とIPv4のヘッダを比べると、IPv4とIPv6が直接的な互換性がないまったく別のプロトコルであることが改めてわかります。比較のために、図4.3にIPv4ヘッダのフォーマットを示します。

---

†6　RFC 2675：D. Borman, S. Deering, R. Hinden, “IPv6 Jumbograms”, 1999年8月
†7　RFC 4294：J. Loughney, “IPv6 Node Requirements”, 2006年4月
†8　RFC 6434：E. Jankiewicz, J. Loughney, T. Narten, “IPv6 Node Requirements”, 2011年12月
†9　RFC 8504：T. Chown, J. Loughney, T. Winters, “IPv6 Node Requirements”, 2019年1月
†10　RFC 2464：M. Crawford, “Transmission of IPv6 Packets over Ethernet Networks”, 1998年12月

CC BY NC SA



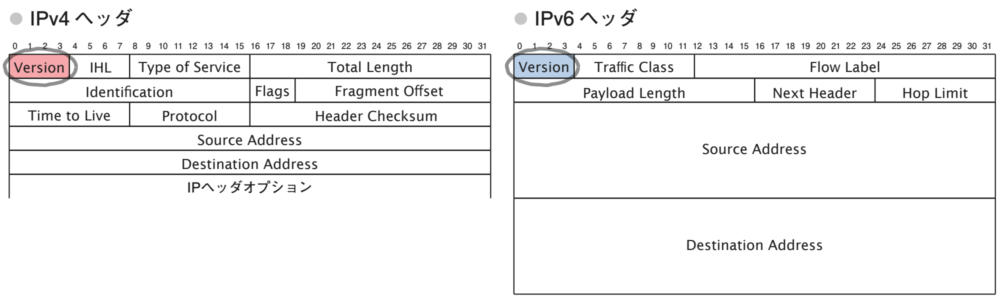
▶ 図4.3　IPv4ヘッダフォーマット

IPv6ヘッダのフォーマットは、バージョンフィールド以外、すべてがIPv4ヘッダのフォーマットとは異なります。フィールドの個数も、IPv4ヘッダでは12個だったのに対し（IPヘッダオプションを含めない）、IPv6ヘッダでは8個に減っています。IPv6ヘッダのほうがヘッダ自体はシンプルになっているといえます。

フィールドの個数だけではなく、IPアドレス自体を含めなければ、フィールドの長さも短縮されています。IPv4ヘッダ（IPヘッダオプションを含めない）では、IPv4アドレス以外の部分は12オクテットでした。一方IPv6ヘッダでは、IPv6アドレス以外の部分が8オクテットなので、IPアドレス自体を含まないヘッダ領域はIPv4ヘッダよりも小さくなっています。ただし、IPv6アドレスはIPv4アドレスより4倍長いので、IPv6ヘッダ自体はIPv4ヘッダ（IPヘッダオプションを含めない）の倍の長さになります。

■ Versionフィールド

まずは、唯一変更されていないフィールドを見てみましょう。ヘッダの各フィールドのうち、IPv4からまったく変更されていないのは、インターネットプロトコルのバージョン番号を示す4ビットのVersionフィールドのみです。フィールドの値としては、IPv4では4を示す0100、IPv6では6を示す0110という違いはありますが、フィールド名称とビット数および用途は変更されていません。IPパケットを処理する機器は、先頭4ビットのVersionフィールドを確認することで、そのパケットがIPv4なのかそれともIPv6なのかを判断します。



▶ 図4.4　IPv4ヘッダとIPv6ヘッダのVersionフィールド

IPv5は？

IPv4の次のインターネットプロトコルがIPv6なのは、「L3プロトコルとしてIPv4の次にどのようなものを作るのか」という競争の結果としてIPv6が勝ち残ったからです。この競争の残骸として5、7、8、9という数値が確保されたまま残っています。

IPv4とIPv6それぞれのヘッダにおける先頭4ビットはVersionフィールドです。このフィールドにはインターネットプロトコルの「バージョン番号」が格納され、そこで利用可能な値はIANAで管理されています。IANAで公開されている情報[†11]を見ると、インターネットプロトコルのバージョン番号として、4と6だけではなく5、7、8、9という値が“Reserved(Historic)”という形で確保されています。IPv4の次のバージョンがIPv6であり、IPv5が存在していないのは、このバージョン番号の値として5がすでに別の用途に利用されていたからです。

バージョン番号の5という値を利用するのは、1990年に発行されたRFC 1190で定義されているST-IIというプロトコルです。ST-IIは、IPv4やIPv6のようなインターネットプロトコルではありませんが、インターネットプロトコル同様に先頭にSTヘッダという名前のヘッダがあります。そして、このSTヘッダの最初の4ビットを「0101」（十進数で表記すると5）とすることで、IPv4ヘッダとSTヘッダを区別できるようにしてあります。ST-IIがヘッダの先頭4ビットのバージョンフィールドで5という値を使うので、それと競合する5という値はインターネットプロトコルのバージョンとしては使わず、IPv4の次は6というバージョン番号が採用されました。ST-IIは実験的なプロトコルという位置づけでRFC化されていますが、その後ST-IIが普及せず、ヘッダの最初の4ビットで利用する5という値が確保されたまま、今では使われていません。

7という番号は、IPアドレスが64ビットのIPバージョン7に割り当てられています（RFC 1475[†12]）。8という番号は、IPv4を置き換えることを目指したPipというプロトコルのために確保されています（RFC 1621[†13]）。9という番号は、TUBAというプロトコルのために確保されています（RFC 1347[†14]）。考えにくいことではありますが、万が一、IPv6の次世代版のインターネットプロトコルが作られるとすれば、そのためにバージョン番号として次に空いている値は10なので、IPv10になるのかも知れません。

ちなみに、IPv4より前のバージョン番号はRFC 755で知ることができます。このRFC 755[†15]には、“version of the internetwork general protocol”として、4ビットのフィールドで使う値として0から4、および15に関する記述もあります。具体的には、0を“March 1977 version”、1を“January 1978 version”、2を“February 1978 version A”、3を“February 1978 version B”、4を“February 1979 version 4”としています。15は“Reserved”です。

---

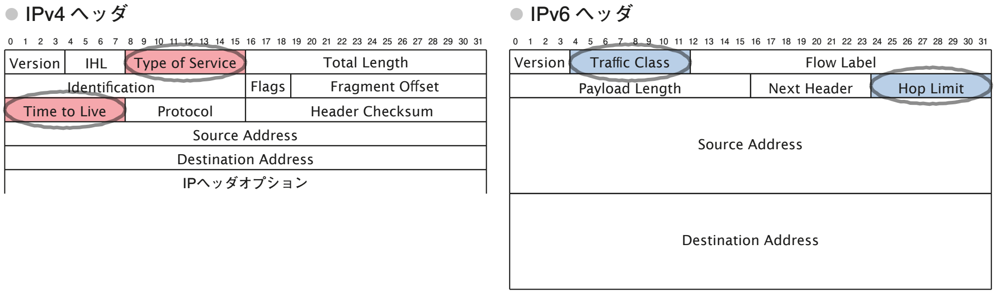
†11　https://www.iana.org/assignments/version-numbers/version-numbers.xhtml
†12　RFC 1475：R. Ullmann, “TP/IX: The Next Internet”, 1993年6月
†13　RFC 1621：P. Francis, “Pip Near-term Architecture”, 1994年5月
†14　RFC 1347：R. Callon, “TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing”, 1992年6月
†15　RFC 755：J. Postel, “Assigned numbers”, 1979年5月

CC BY NC SA

■ 名称が変更されたフィールド（Hop Limit と Traffic Class）

IPv6 ヘッダの Hop Limit フィールドと Traffic Class フィールドは、それぞれ IPv4 ヘッダの TTL フィールドと ToS フィールドと同じ用途であり、名称だけが変更された部分だといえます。

これらの名称変更は、実状に沿っていなかった定義を正しいものへと直した結果です。特に IPv4 ヘッダの TTL フィールドは、Time To Live という名称のとおりパケットが生存し続けられる「時間」を表しており、IPv4 を規定している RFC 791 では単位を「秒」としていますが、一方では「パケットを転送するたびに最低でも 1 は TTL の値を減算すること」が求められており、現実の実装は秒数ではなくホップ数を表している状態でした。そのため IPv6 ヘッダでは、同じ用途のフィールドの名称が Hop Limit フィールドと定義し直されています。

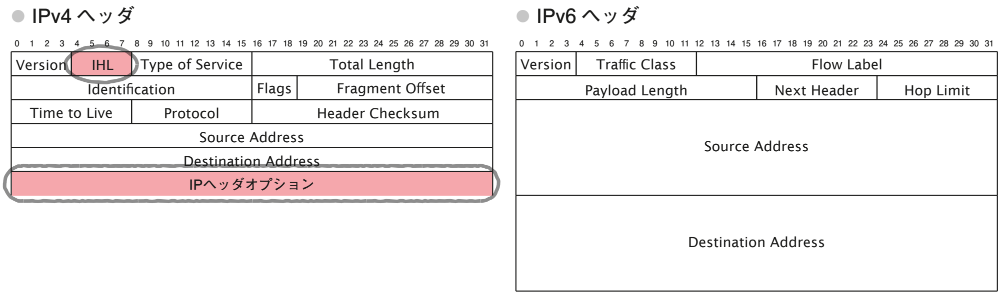


▶ 図 4.5　ToS と TTL（IPv4 ヘッダ）、および、Traffic Class と Hop Limit（IPv6 ヘッダ）

■ 可変長から固定長への変更に伴って再定義されたフィールド

IPv4 ヘッダから IPv6 ヘッダへの変更で特に大きいのは、ヘッダの長さが固定長になったことです。IPv4 には IP ヘッダオプションという仕組みがあり、IPv4 ヘッダは可変長でした。IPv4 ヘッダには、その長さを示すための IHL（Internet Header Length）フィールドがありました。この IHL フィールドで IP ヘッダオプションの長さおよび有無を知ることができます。

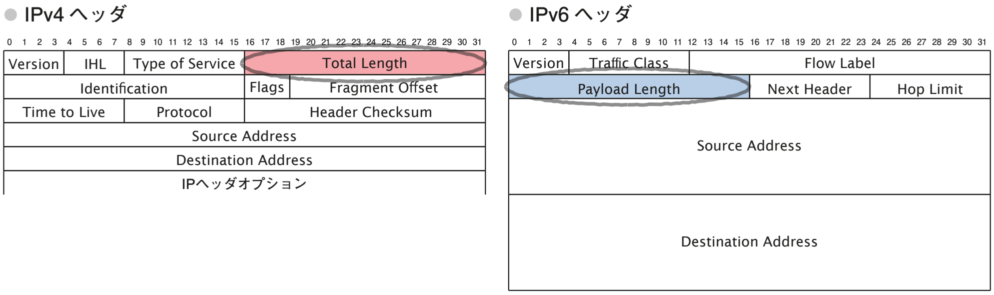
これに対し、IPv6 ヘッダは 40 オクテットの固定長です。IPv4 のように IP ヘッダの長さを表現する必要がなくなったことから、IPv6 では IHL（Internet Header Length）フィールドが削除されました。



▶ 図 4.6　IPv4 ヘッダにおける IP ヘッダオプションと IHL は IPv6 ヘッダにはない

IPv6 ヘッダが固定長になったことに伴い、パケットの長さを示すフィールドの性格も、IPv4 と IPv6 とでは少し異なっています。IPv4 ヘッダで Total Length フィールドが果たしていた役

割は、IPv6ヘッダではPayload Lengthフィールドが果たしていますが、IPv4のTotal Lengthフィールドが IPv4ヘッダを含む長さであったのに対し、IPv6のPayload Lengthフィールドは IPv6ヘッダを含まない長さとなっています。

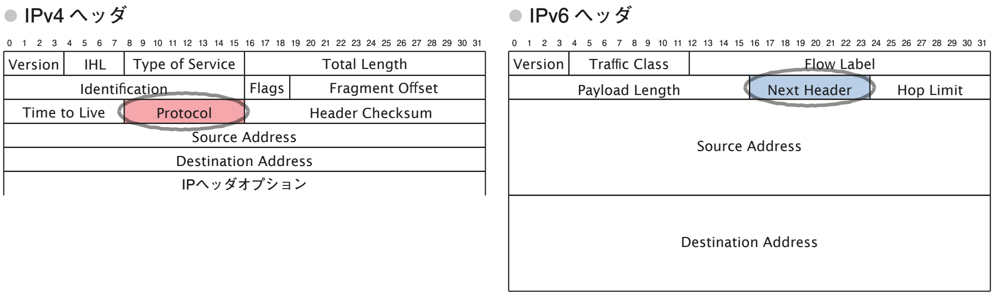


▶図4.7　Total Lengthフィールド（IPv4ヘッダ）とPayload Lengthフィールド（IPv6ヘッダ）

“payload”（ペイロード）は、もともとは貨物などで「お金を払って運んでもらう部分」を指す単語です（“pay”には「お金を払う」、“load”には「積荷」という意味があります）。コンピュータネットワークの世界では、このペイロードという用語によって、「運ばれるデータの本体部分」を表します。IPv4ヘッダは可変長なので、パケットの全体の長さ（total length）をヘッダに格納していましたが、IPv6ではヘッダが固定長なので、積荷の長さ（payload length）をヘッダに格納するようになっています。

■ Protocolフィールドと Next Headerフィールド

Next Headerフィールドは、IPv4ヘッダにおけるProtocolフィールドの名称が変更されただけのように思われがちですが、実際には役割も大きく変わっています。たとえば、Next Headerフィールドは IPv6でのフラグメントでも利用されます。また、IPv6拡張ヘッダ（IPv4におけるIPヘッダオプションの役割を担う）でもNext Headerを利用します。つまりNext Headerフィールドは、IPv4のProtocolフィールドが担っていた役割だけではなく、他のさまざまな役割も担うフィールドに変わっているのです。



▶図4.8　Protocolフィールド（IPv4ヘッダ）とNext Headerフィールド（IPv6ヘッダ）

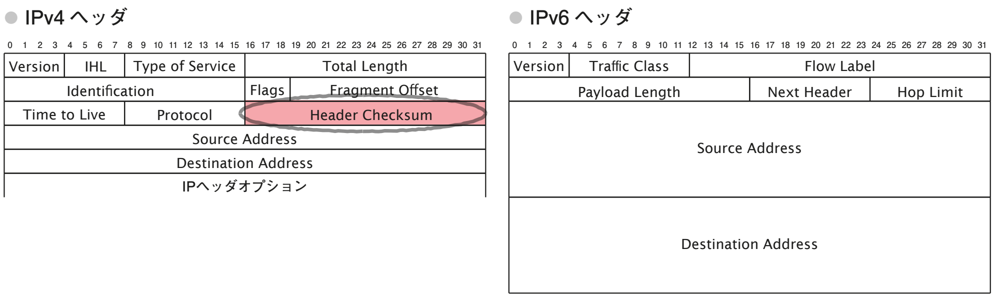
Next Headerフィールドは、文字どおり「次のヘッダ」を表します。たとえば、IPv6ヘッダの直後にTCPヘッダが続く場合、Next HeaderフィールドはTCPを示す6になります。IPv6ヘッダの直後にくるのは、IPv6パケットのペイロードに含まれるTCPヘッダであり、TCPヘッ

ダも「ヘッダ」なので、「Next Header」でそれを示すわけです。一方、IPv6 ヘッダの後に続くのが IPv6 拡張ヘッダの場合には、その IPv6 拡張ヘッダを示す値が Next Header フィールドに入ります。たとえば、IPv6 ヘッダの後にルーティングヘッダ（IPv6 拡張ヘッダの一種）が続く場合、Next Header フィールドの値はルーティングヘッダを表す 43 になります。

Next Header フィールドによって、さまざまな拡張機能を IPv6 ヘッダの外に出せることは、IPv6 ヘッダを固定長にすることにも貢献しています。さらに、IPv6 拡張ヘッダが存在しない場合にも「次のヘッダ」を示すようにすることで、IPv4 ヘッダの IHL フィールドと Protocol フィールドを融合して余分なフィールドを削減しているとも考えられます。

■ チェックサムを格納するフィールド

IPv4 ヘッダにあった Header Checksum フィールドも IPv6 ヘッダでは削除されています。これは、IPv4 の開発当時と比べて回線の品質が向上したことに加え、「TCP や UDP などの上位層でチェックサムによる誤り検出を行うべきである」という考えが採用されたからです。上位層でのチェックサム計算については 4.2 節で説明します。なお、IPv4 ではパケットを転送するたびに TTL を減らしてチェックサムを再計算する仕組みだったので、そのためのオーバーヘッドがありましたが、IPv6 ではその必要がなくなり、そのぶんだけ転送処理にかかる計算も減らせるようになっています。



▶ 図 4.9　IPv4 ヘッダにおける Header Checksum フィールドは IPv6 ヘッダにはない

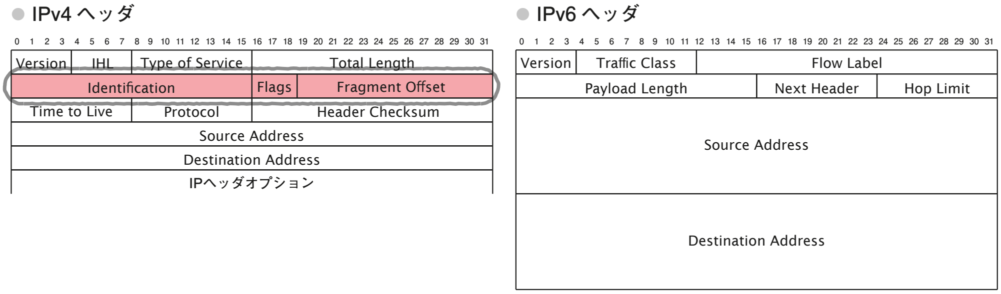
■ フラグメンテーションに関連するフィールド

IPv6 では経路途中でのフラグメンテーションが廃止され、送信元ノードのみがフラグメント化を行うようになりました。それに伴い、IPv4 ヘッダに存在していた Identification、Flags、Fragment Offset の各フィールドも削除されています。

IPv6 ヘッダにはフラグメンテーションに関連するフィールドがありませんが、IPv6 でフラグメンテーションがなくなったわけではありません。IPv6 におけるフラグメンテーションでは、関連する情報が IPv6 拡張ヘッダに格納されます。IPv6 のフラグメンテーションについては第 9 章で詳しく解説します。

■ フローラベル

IPv6 ヘッダには Flow Label というフィールドがあります。IPv4 ヘッダには含まれていなかった、IPv6 で新たに追加されたフィールドです。



▶ 図4.10　Identification、Flags、Fragment Offsetフィールドは IPv6ヘッダにはない

Flow Labelフィールドに格納される**フローラベル**とは、それまで「送信元IPアドレス」と「宛先IPアドレス」、「送信元ポート番号」と「宛先ポート番号」、それに「トランスポートプロトコルの種類」という5つの情報をもとに識別されていた通信の**フロー**を明示的に識別するためのラベルです。従来の5つの情報では、暗号化された通信においてフローを識別できないことがあり、それに代わって利用できる識別子として、このフィールドでフローラベルを用いるようなことが想定されていました[†16]。

IPv6でフローラベルをどのように利用するかについては、IPv6の最初のRFCが発行される前の仕様策定の頃から議論が続いていました。IPv6ヘッダの仕様を規定していたRFC 2460[†17]では、フローラベルを実験的なものであるとしつつ、「フローラベルによってフロー識別を可能とし、IPv6ルータによる特別な処理を実現可能にする」ものであるとしています。しかし、RFC 2460を上書きして廃止したRFC 8200では、「複数のパケットを単一のフローとして識別するための20ビットのFlow Labelフィールド」というシンプルな説明になっており、フローラベルについて詳しい記述はありません。フローラベルそのものに関しては、RFC 6437[†18]を参照する形に変わっています。

そのRFC 6437で示されているIPv6フローラベルの仕様は、2004年に発行されたRFC 3697[†19]を2011年に廃止して置き換えられたものです。RFC 3697とRFC 6437の違いとしては、たとえば「フローラベルの値を設定できるノード」の変化などが挙げられます（他の相違点はRFC 6436[†20]でまとめられています）。RFC 3697では、フローラベルの値を設定できるのは送信元ノードだけであり、途中経路においてフローラベルの値を変更することは禁止されていました。しかし、それではフローラベルをルーティングなどと組み合わせて使うことが難しくなってしまうので、その要件が緩和され、どのノードであってもフローラベルの値を変更できるようになりました。

IPv6の仕様が議論されていた1990年代と今では、ネットワークをめぐる環境が大きく変

†16　ただし、IPsecなどで暗号化された通信に対してフローラベルを使ってしまうと、攻撃者が通信内容を解析するためのヒントを与えてしまう可能性もあります。
†17　RFC 2460：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 1998年12月
†18　RFC 6437：S. Amante, B. Carpenter, S. Jiang, J. Rajahalme, “IPv6 Flow Label Specification”, 2011年11月
†19　RFC 3697：J. Rajahalme, A. Conta, B. Carpenter, S. Deering, “IPv6 Flow Label Specification”, 2004年3月
†20　RFC 6436：S. Amante, B. Carpenter, S. Jiang, “Rationale for Update to the IPv6 Flow Label Specification”, 2011年11月

わってしまっています。現在では、TCPやUDPのポート番号だけではなく、それらが運ぶペイロードの中身まで解析して制御を行うDPI（Deep Packet Inspection）が珍しいものではなくなりました。もしフローラベルを有効活用できれば、そうしたIPv6よりも上位にある層の情報を調べることなくさまざまな処理が可能になり、途中ノードにおける処理の負荷を大幅に軽減することが期待されます。

しかし、実際に運用されているネットワークへの導入が現実的かどうかはまったく別の話です。今後、IPv6においてフローラベルおよびFlow Labelフィールドがどのように利用されていくのか、本書執筆時点ではよくわかっていません。IPv6フローラベルの用途については、RFC 6294[†21]にこれまでの提案などがまとめられているので、参考にしてください。

4.2　上位層でのチェックサム計算に使う仮想ヘッダ

チェックサム（checksum）は、誤り検出符号の一種です。インターネットでは、パケットの伝送時にエラーが発生した場合にそれを検出することを目的として、さまざまな場面でチェックサムが利用されています。

インターネットで使われるチェックサムは、ヘッダなどのチェックサム用フィールドに初期設定（0をセットするなど）を施した状態で、チェックサムとして計算する範囲を16ビット単位で加算し、その加算した結果を16ビットにしたうえで、その1の補数を求めるという方法で計算します。チェックサム計算としては簡易な方式なので、伝送時のエラーを検出できないこともありえます。この計算方法については、RFC 1071[†22]で具体的に解説されています。

前述したようにIPv6自体のヘッダにはチェックサム用のフィールドがありませんが、上位層であるTCPやUDPのヘッダにはチェックサム用フィールドがあります。これらの上位層プロトコルの基本仕様を示したRFC 793[†23]およびRFC 768[†24]では、IPヘッダにしか含まれない送信元アドレス情報や宛先アドレス情報で発生したエラーを検出するため、TCPおよびUDPにおける16ビットのチェックサム計算において図4.11のような**疑似ヘッダ**（pseudo header）を含めることとされています。

0	1	2	3	4	5	6	7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Address									
Destination Address									
0	0	0	0	0	0	0	0	プロトコル番号（TCP：6、UDP：17）	TCP/UDPパケット（ヘッダ＋データ）の長さ

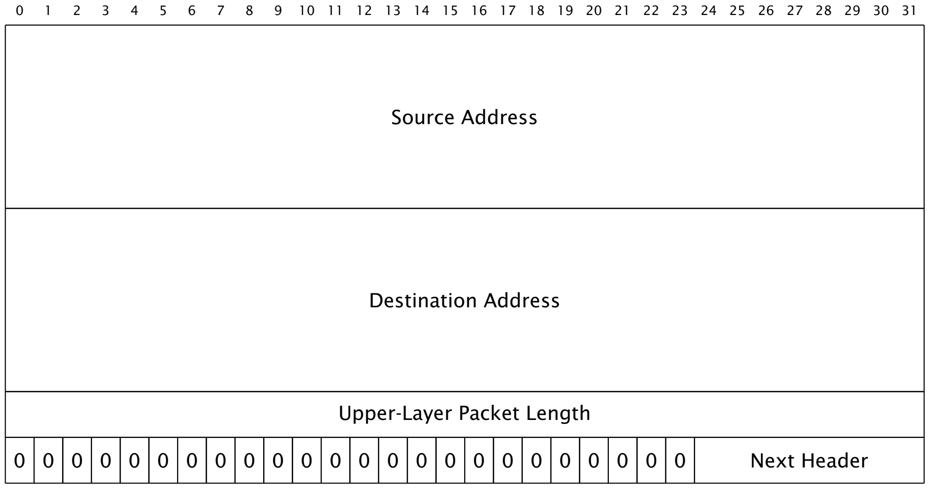
▶ 図4.11　RFC 768およびRFC 793における疑似ヘッダフォーマット

図4.11からわかるように、TCPとUDPの仕様で定義されている疑似データはIPアドレスが32ビットであることを前提としています。そのため、IPv6でTCPやUDPを使う場合には、この方法で疑似ヘッダを使うことができません。そこで、RFC 8200では、IPv6にとって上位層

[†21] RFC 6294：Q. Hu, B. Carpenter, “Survey of Proposed Use Cases for the IPv6 Flow Label”, 2011年6月
[†22] RFC 1071：R.T. Braden, D.A. Borman, C. Partridge, “Computing the Internet checksum”, 1988年9月
[†23] RFC 793：J. Postel, “Transmission Control Protocol”, 1981年9月
[†24] RFC 768：J. Postel, “User Datagram Protocol”, 1980年8月

(cc) BY NC SA

にあたるTCPやUDPなどのプロトコルでのチェックサム計算で使うための**疑似IPv6ヘッダ**が定義されています。



▶図4.12　疑似IPv6ヘッダフォーマット

IPv6では、ICMPv6でもこの疑似ヘッダが使われます。IPv4のICMPでは、ICMPヘッダのみでチェックサムの計算が行われましたが、IPv6ヘッダにチェックサムフィールドがないこともあり、ICMPv6ではTCPとUDP同様に疑似IPv6ヘッダとICMPv6ヘッダを対象としたチェックサム計算が行われるのです。

4.2.1　IPv6でのUDPチェックサムについて

UDPヘッダには、16ビットのChecksumフィールドがあります。UDPの仕様を記載しているRFC 768では、UDPヘッダのChecksumフィールドにゼロを入れること（ゼロチェックサム）が許可されていました。Checksumフィールドがゼロの場合には、送信側でチェックサムを生成しなかったと解釈するように明記されています。チェックサムが生成されない例としては、デバッグ目的の場合や、UDPを利用する上位層のプロトコルでチェックサムを必要としない場合が挙げられています。

実際、IPv4では、UDPを使ってトンネリングを行うアプリケーションや機器で、Checksumフィールドをゼロにするという運用が行われています。トンネリングを行うとき、UDPパケットのペイロード部分にIPヘッダやTCPヘッダなどが含まれることも多く、トンネルを実現しているUDPパケットでチェックサムの計算を行うことが必ずしも必要ではないからです。UDPでは、IPヘッダに含まれるIPアドレスなどを計算に利用するための疑似ヘッダ、UDPヘッダ、そしてペイロード部分に対してチェックサム計算を実施します。ヘッダだけではなく、パケットのペイロード部分を含めて計算するので、チェックサムの計算ではパケット全体を解析する必要があります。UDPチェックサムの計算を行うと、パケットのサイズが大きくなったとき、メモリやCPUなどの計算機資源をどうしても消費してしまいます。トンネリングのためのアプリケーションや機器では、UDPチェックサムの計算をしないようにすることで効率化を図っているのです。

ただし、UDPヘッダのChecksumフィールドでゼロが許可されるのは、IPv4の話です。IPv6ではIPv6ヘッダにおけるチェックサムが廃止されたことに伴い、TCPとUDPでチェックサムを計算することが必須となりました。1995年に発行されたIPv6の最初の基本仕様であるRFC 1883と、RFC 1883を上書き廃止するRFC 2460でも、IPv4とは異なりIPv6ではUDPチェックサムの計算が必須であることが明記されています。

IPv6においてUDPヘッダのChecksumフィールドにゼロを使えないと、UDPを使ってトンネリングを行うアプリケーションや機器の動作効率が悪くなってしまいます。そのため、IPv6でもIPv4と同様にUDPのゼロチェックサムを求める声が根強くありました。

UDPのゼロチェックサム禁止は、2013年にRFC 6935[†25]とRFC 6936[†26]によって条件付きで緩和されています。

RFC 2460を上書き廃止したIPv6の基本仕様であるRFC 8200も、RFC 6935とRFC 6936の内容を反映したものになっているので、UDPでのチェックサムは基本的には必要ですがゼロチェックサムの例外が認められています。具体的には、UDPにおけるチェックサム計算の例外として、「UDPを使ったカプセル化によるトンネリングを行う場合にはゼロチェックサムを使ってもよい」としています。ただし、デフォルト機能としてはUDPヘッダのChecksumフィールドがゼロのパケットを破棄しつつ、「特定のポートではゼロチェックサムを例外的に扱えるような仕組みになっていること」などが条件とされています。IPv4とは異なり、IPv6では基本的にはUDPでのチェックサムは必須であり、トンネリングを行う場合のみという条件付きでゼロチェックサムが許容されている点に注意が必要です。

4.3 IPv6拡張ヘッダ

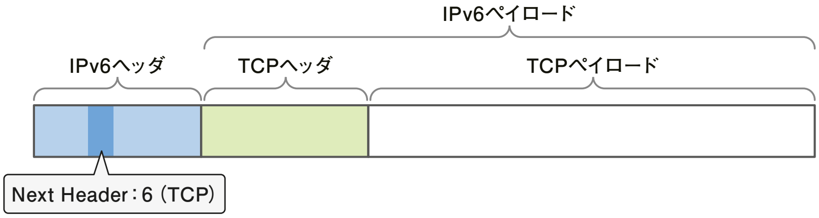
IPv6拡張ヘッダは、IPv6ヘッダの機能を拡張できるようにするための仕組みであり、IPv6の大きな特徴になっています。IPv6で拡張ヘッダのような仕組みが必要になるのは、IPv6ヘッダが固定長であり、IPv4ヘッダにあるような可変長のオプションフィールドがないためです。IPv6ヘッダを固定長にしつつも、さまざまな機能を持たせることを実現しているのが、IPv6拡張ヘッダなのです。

IPv6パケットは0個もしくは複数個のIPv6拡張ヘッダを持てます。つまり、IPv6拡張ヘッダはIPv6パケットに必須の要素ではなく、設定する場合は1つのIPv6パケットに対して複数のIPv6拡張ヘッダを追加できます。

図4.13に、IPv6拡張ヘッダを利用しない場合のIPv6パケットの例を示します。この場合は、IPv6ヘッダの次にTCPヘッダが直接付属し、IPv6ヘッダのNext Headerフィールドには TCPを示すプロトコル番号の「6」という数値が入ります。TCPヘッダの後ろにはTCPペイロードが続きます。

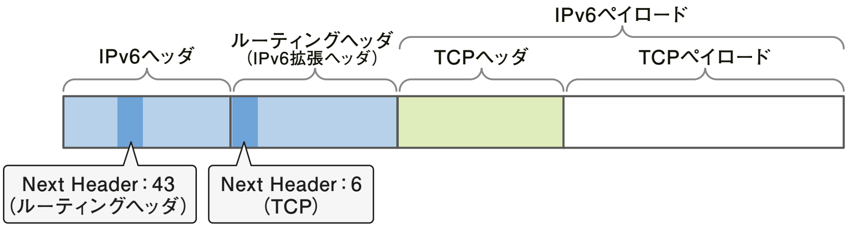
†25 RFC 6935：M. Eubanks, P. Chimento, M. Westerlund, “IPv6 and UDP Checksums for Tunneled Packets”, 2013年4月

†26 RFC 6936：G. Fairhurst, M. Westerlund, “Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums”, 2013年4月



▶ 図4.13　IPv6拡張ヘッダなし

次は、IPv6拡張ヘッダを含むIPv6パケットの例です。図4.14に、IPv6拡張ヘッダの1つであるルーティングヘッダを含むTCPパケットの例を示します。この場合、IPv6ヘッダのNext Headerフィールドには、IPv6ルーティングヘッダを示すプロトコル番号の「43」という数値が入ります。さらに、IPv6ルーティングヘッダの中にあるNext Headerフィールドには、TCPを示すプロトコル番号の「6」という数値が入ります。



▶ 図4.14　IPv6拡張ヘッダ例（ルーティングヘッダ）

このように、IPv6ヘッダには、次にIPv6拡張ヘッダが続くこともあれば、TCPのような上位層のプロトコルヘッダが続くこともあります。IPv6拡張ヘッダは複数指定できるので、IPv6ヘッダの次にIPv6拡張ヘッダが続く場合には、そのIPv6拡張ヘッダの次にさらに別のIPv6拡張ヘッダが続くこともあります。IPv6では、このような仕組みを実現するために、IPv4における「Protocol」フィールドが「Next Header」とされています。文字どおり「次のヘッダ」というわけです。

Next Headerフィールドに指定するプロトコル番号はIANAが管理しており、IANAのサイト[†27]で一覧を参照できます。そのうちIPv6拡張ヘッダを示す番号を表4.1にまとめます。

---

†27　“Protocol Numbers”：
https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

CC BY NC SA

▶ 表4.1 IPv6拡張ヘッダのプロトコル番号

番号	プロトコル	参照
0	Hop-by-Hop オプションヘッダ	RFC 8200
43	ルーティングヘッダ	RFC 8200、RFC 5095[†28]
44	フラグメントヘッダ	RFC 8200
50	Encapsulating Security Payload ヘッダ	RFC 4303[†29]
51	認証ヘッダ	RFC 4302[†30]
60	宛先オプションヘッダ	RFC 8200
135	モビリティヘッダ	RFC 6275[†31]
139	Host Identity Protocol	RFC 7401[†32]
140	Shim6 Protocol	RFC 5533[†33]
253	実験やテスト用	RFC 3692[†34]、RFC 4727[†35]
254	実験やテスト用	RFC 3692、RFC 4727

IPv6拡張ヘッダを解釈するのは、基本的にIPv6ヘッダの宛先フィールドに記載されているノードのみです。途中の経路上にあるノードでは、IPv6拡張ヘッダを処理しません。それどころか、途中の経路上にあるルータが転送するIPv6パケットに対してIPv6拡張ヘッダを新たに追加したり、IPv6パケットに含まれているIPv6拡張ヘッダを削除したりすることは禁止されています。ただし、途中の経路上の各ホップに存在するノードでも、Hop-by-Hopオプションだけは処理します。

NOTE

経路の途中のノードではIPv6拡張ヘッダを解釈しないとはいえ、TCPなどの上位プロトコルを経路の途中で参照して処理するDPI（Deep Packet Inspection）機器などでは、IPv6拡張ヘッダが多いパケットについては処理が遅くなる可能性はあります。
IPv6拡張ヘッダの削除はRFC 8200で禁止されていますが、実験的なRFCであるRFC 4782[†36]には、Hop-by-Hopオプションを削除するという記述が含まれています。ただし、RFC 4782で示されているQuick-Start for TCP and IPは、普及している仕様ではありません。
その他、ファイアウォールにおけるIPv6拡張ヘッダの扱いについてはRFC 7045[†37]を参照してください。

†28 RFC 5095：J. Abley, P. Savola, G. Neville-Neil, “Deprecation of Type 0 Routing Headers in IPv6”, 2007年12月
†29 RFC 4303：S. Kent, “IP Encapsulating Security Payload (ESP)”, 2005年12月
†30 RFC 4302：S. Kent, “IP Authentication Header”, 2005年12月
†31 RFC 6275：C. Perkins, D. Johnson, J. Arkko, “Mobility Support in IPv6”, 2011年7月
†32 RFC 7401：R. Moskowitz, T. Heer, P. Jokela, T. Henderson, “Host Identity Protocol Version 2 (HIPv2)”, 2015年4月
†33 RFC 5533：E. Nordmark, M. Bagnulo, “Shim6: Level 3 Multihoming Shim Protocol for IPv6”, 2009年6月
†34 RFC 3692：T. Narten, “Assigning Experimental and Testing Numbers Considered Useful”, 2004年1月
†35 RFC 4727：B. Fenner, “Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers”, 2006年11月
†36 RFC 4782：S. Floyd, M. Allman, A. Jain, P. Sarolahti, “Quick-Start for TCP and IP”, 2007年1月
†37 RFC 7045：B. Carpenter, S. Jiang, “Transmission and Processing of IPv6 Extension Headers”, 2013年12月

(cc) BY NC SA

IPv6パケットを受信する側では、IPv6拡張ヘッダを、先頭にあるものから順次処理することになっています。すべてのIPv6拡張ヘッダを一通りスキャンしてから任意の順番で処理することは禁止されています。そのため、IPv6拡張ヘッダの順番には大きな意味があります。RFC 8200では、1つのIPv6パケットに複数のIPv6拡張ヘッダを含める場合の順番を次のように推奨しています。

1. IPv6ヘッダ
2. Hop-by-Hopオプションヘッダ
3. 宛先オプションヘッダ（IPv6ヘッダの宛先フィールドに記載された最初の宛先と、ルーティングヘッダに記載されているその他の宛先で処理されるべきもの）
4. ルーティングヘッダ
5. フラグメントヘッダ
6. 認証ヘッダ
7. Encapsulating Security Payloadヘッダ
8. 宛先オプションヘッダ（宛先ノードのみで処理されるべきもの）
9. 上位層のヘッダ

なお、宛先オプションヘッダを除き、同じ種類のIPv6拡張ヘッダが1つのIPv6パケットに複数含まれることはありません。

以降では、RFC 8200で示されているIPv6拡張ヘッダのフォーマットをいくつか紹介します。フラグメントヘッダについては9.2節で、IPsecで使う認証ヘッダとEncapsulating Security Payloadヘッダに関しては14.8節で、それぞれ紹介します。

4.3.1 Hop-by-Hopオプションヘッダ

Hop-by-Hopオプションヘッダは、パケットを転送する各ルータで何らかの処理をするときに必要となる情報を格納する拡張ヘッダです。

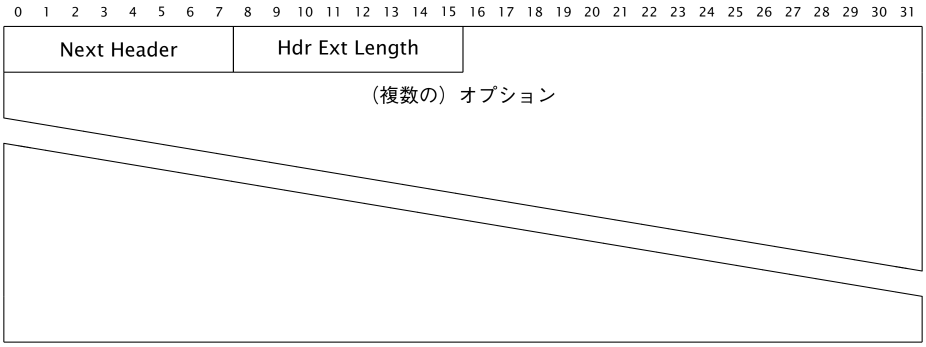
Hop-by-Hopオプションヘッダのプロトコル番号は「0」です。IPv6拡張ヘッダが付く場合には、Hop-by-Hopオプションヘッダを IPv6ヘッダの直後に続けることが推奨されているので、IPv6拡張ヘッダのNext Headerフィールドで「0」が使われることはありません。

以前は、Hop-by-Hopオプションはすべてのルータによって処理されることが前提でした。IPv6の旧基本仕様であるRFC 2460（RFC 8200によって廃止）では、Hop-by-Hopオプションは、そのパケットが転送されていく経路上のすべてのルータによって処理される必要があるとしています。しかし、最新の基本仕様であるRFC 8200では、Hop-by-Hopオプションを処理するように設定されたルータのみがそれを処理するという内容に変わっています。これは、RFC 7045で行われたRFC 2460に対するアップデートをRFC 8200に反映させた結果です。

Hop-by-Hopオプションヘッダのフォーマットを図4.15に示します。

格納できるオプションについては4.3.3項で説明します。1つのHop-by-Hopオプションヘッダに、複数のオプションが含まれている場合もありますが、それら複数のオプション全体をノードであらかじめスキャンしてから処理することは禁止されています。複数のオプションを格納する場合には、ノードで処理される順番に並べる必要があります。

(cc) BY NC SA



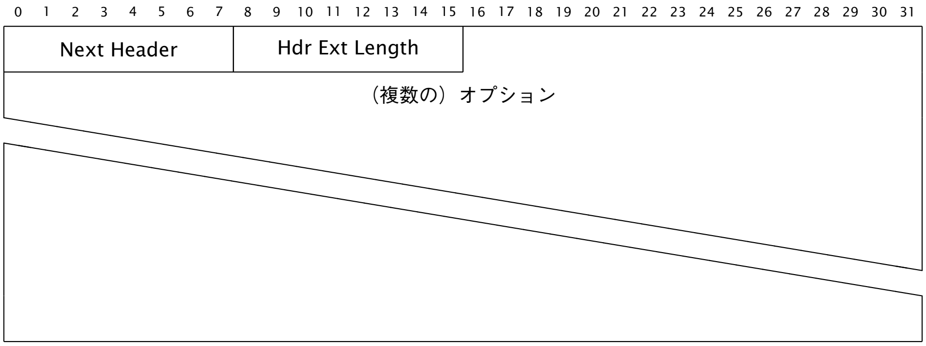
▶ 図4.15　Hop-by-Hop オプションヘッダフォーマット

4.3.2　宛先オプションヘッダ

宛先オプションヘッダは、パケットの宛先（終点）のみで処理される内容を記述するためのIPv6拡張ヘッダです。宛先オプションヘッダは、IPv6ヘッダおよびIPv6拡張ヘッダのNext Headerフィールドでは「60」という値で表現されます。

宛先オプションヘッダは、他のIPv6拡張ヘッダとは異なり、1つのIPv6パケットに最大2個含まれることがあります。

宛先オプションヘッダのフォーマットを図4.16に示します。



▶ 図4.16　宛先オプションヘッダフォーマット

格納できるオプションについては4.3.3項で説明します。1つの宛先オプションヘッダに複数のオプションが含まれている場合もありますが、それら複数のオプション全体をノードであらかじめスキャンしてから処理することは禁止されています。複数のオプションを格納する場合には、ノードで処理される順番に並べる必要があります。

宛先オプションヘッダは、IPv6ヘッダに示された宛先へパケットが到達した際に行われてほしい処理を指定するのに使います。そのような処理であっても、フラグメントヘッダと認証ヘッダの2種類については、宛先オプションヘッダとは別のIPv6拡張ヘッダとして定義されています。

CC BY NC SA

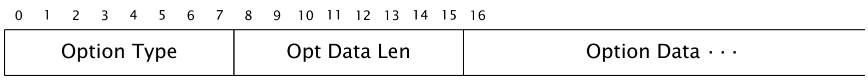
4.3.3 Hop-by-Hopオプションヘッダと宛先オプションヘッダで使われるTLV形式

Hop-by-Hopオプションヘッダと宛先オプションヘッダには、オプションとしていくつかの付加的な情報を格納できるようになっています。これらの付加的な情報は、**TLV形式**で格納します。

NOTE

TLV形式は、Hop-by-Hopオプションヘッダと宛先オプションヘッダに限らず、さまざまな通信プロトコルでデータの格納に利用される一般的なデータ形式です。省略可能なデータの種類（Type）、そのデータの長さ（Length）、実際のデータの値（Value）がセットになった構造であることから、頭文字をとって「TLV形式」と総称されています。

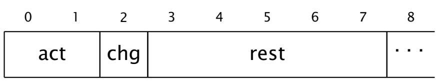
RFC 8200では、これらのオプションヘッダ内に格納するオプションのTLV形式について、図4.17のように定義しています。



▶ 図4.17　Hop-by-Hopオプションヘッダと宛先オプションヘッダで使われるTLV形式

- Option Type

  オプションの種類を表すフィールドです。IANAのサイト[†38]で一覧を確認できます。このフィールドの内部は、さらに以下のように区分されています。



先頭の2ビット（図中のact）は、ノードがOption Typeに未対応のときのアクションを表しています。

- 00：このオプションを無視する
- 01：このパケットを破棄する
- 10：このパケットを破棄し、ICMPv6のParameter Problemメッセージ（5.2.4項参照）を送信元に返す。その際、Codeフィールドには、未知のオプションに遭遇したことを示す2を設定する

[†38] “Destination Options and Hop-by-Hop Options”：https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml#ipv6-parameters-2

- 11：このパケットを破棄し、宛先アドレスがマルチキャストでない場合のみ、ICMPv6のParameter Problemメッセージ（5.2.4項参照）を送信元に返す。その際、Codeフィールドには、未知のオプションに遭遇したことを示す2を設定する

3ビットめのchgは、宛先までの途中経路上でOption Dataの内容変更が発生するかどうかを示します。0なら変更を伴わないことを表し、1なら伴うことを表します。

restの部分は、オプションの種類ごとに定義されています。先頭3ビットに上記のような意味があるので、この部分のみでOption Typeが決まっているように思えるかもしれませんが、あくまでも8ビット全体でOption Typeとして定義されています。

- Opt Data Len

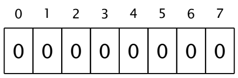
  Option Dataに格納するデータの長さを表します。

- Option Data

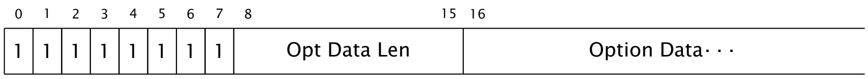
  オプションごとに異なる可変長のフィールドです。

■ パディングのために使うオプションデータ

RFC 8200では、Hop-by-Hopオプションヘッダと宛先オプションヘッダ用のオプションとして、パディングを行うためのPad1およびPadNというオプションを定義しています。それぞれのフォーマットを図4.18および図4.19に示します。



▶ 図4.18　Pad1 オプション



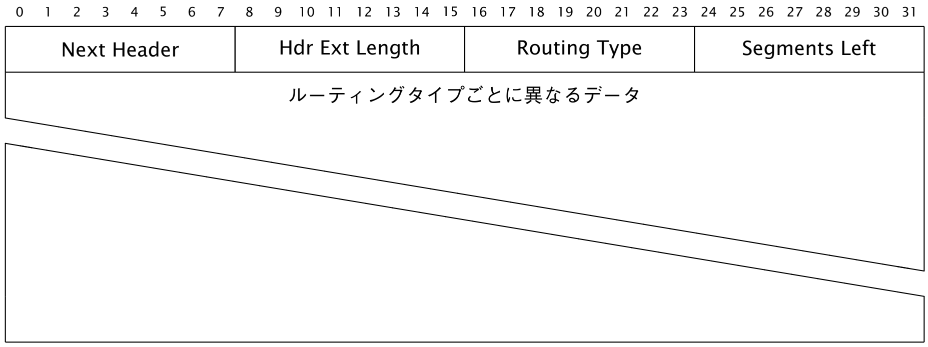
▶ 図4.19　PadN オプション

4.3.4 ルーティングヘッダ

ルーティングヘッダは、IPv6ヘッダに示された宛先IPv6アドレスに到達する間に通過すべきノードを示すためのものです。ルーティングヘッダは、IPv6ヘッダおよびIPv6拡張ヘッダのNext Headerフィールドで「43」という値で表現されます。

ルーティングヘッダのフォーマットを図4.20に示します。ルーティングヘッダにも、Hop-by-Hopオプションヘッダと宛先オプションヘッダで利用されるようなTLV形式のオプションを含められます。

ルーティングヘッダは、Mobile IPv6について規定したRFC 6275と、低電力ネットワーク



▶ 図4.20　ルーティングヘッダフォーマット

でのIPv6の利用について規定したRFC 6554[†39]などでも利用されています。

IANAでは、ルーティングヘッダ内に含まれるRouting Typeフィールドを新たに割り当てする場合のガイドラインを、RFC 5871[†40]として示しています。研究などで実験的にルーティングヘッダを使う場合には、実験的な目的でルーティングヘッダを使う場合のために割り当てられている253および254をRouting Typeフィールドの値として利用可能です。

4.3.5　続くヘッダが存在しない場合

Next Headerフィールドに「59」という値が入っているときは、**No Next Header**を意味し、次に続くヘッダは存在しないものとして扱われます。Next Headerフィールドの値が「59」であるにもかかわらず、IPv6ヘッダのPayload Lengthフィールドの値が「パケットのペイロードにデータが存在している」ことを示している場合、ルータはそのままの状態でIPv6パケットを転送することが求められています。

4.3.6　IPv6拡張ヘッダ仕様の変更

IPv6プロトコルの大きな特徴のひとつである拡張ヘッダですが、現在ではIPv6が設計された当初とは異なる仕様も含まれています。

■ Type 0のルーティングヘッダの廃止

RFC 2460で規定されているType 0のIPv6ルーティングヘッダは、セキュリティ上の理由から、2007年に発行されたRFC 5095によって廃止されています。IPv6ノードが実装すべき機能について規定していたRFC 4294[†41]でも、Type 0のIPv6ルーティングヘッダが必須条件の1つに含まれていましたが、2007年以降は対応不要となっています。それらを反映し、RFC 2460を廃止して上書きするRFC 8200（2017年発行）にも、Type 0のIPv6ルーティングヘッダは含まれていません。

---

†39　RFC 6554：J. Hui, JP. Vasseur, D. Culler, V. Manral, “An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)”, 2012年3月

†40　RFC 5871：J. Arkko, S. Bradner, “IANA Allocation Guidelines for the IPv6 Routing Header”, 2010年5月

†41　2011年に発行されたRFC 6434および2019年に発行されたRFC 8504によって廃止されています。

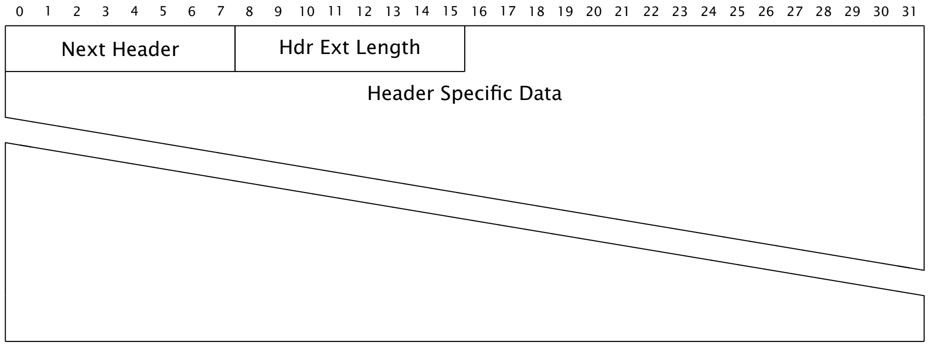
BY NC SA

NOTE

IPv6 ルーティングヘッダそのものが RFC 5095 で廃止されたわけではありません。RFC 5095 が廃止しているのは Type 0 だけであり、Mobile IPv6 が依存する Type 2 の IPv6 ルーティングヘッダなどは引き続き利用可能です。

■ 拡張ヘッダの先頭 16 ビットに対する追加要求

2012 年に発行された RFC 6564[†42] では、すべての IPv6 拡張ヘッダが図 4.21 のフォーマットを満たすように要求しています。すなわち、「IPv6 拡張ヘッダの先頭 8 ビットが Next Header を示し、次の 8 ビットで IPv6 拡張ヘッダの長さを示すようにする」という要求です。



▶ 図 4.21　RFC 6564 が求める IPv6 拡張ヘッダフォーマット

- Next Header（8 ビット）
  この拡張ヘッダの直後にくるプロトコルの種類を示します。IPv6 ヘッダにおける Next Header フィールドと同じく、IANA によって割り当てられている値（IPv4 の Protocol Type フィールドで利用される値と同じ値）を指定します。
- Hdr Ext Len（8 ビット）
  拡張ヘッダの全長を、8 オクテット単位で数えた値を指定します。ただし、最初の 8 オクテットは含みません。
- Header Specific Data（可変長）
  拡張ヘッダの種類によって異なるフィールドです。

ここで問題になるのが、フラグメントヘッダは上記のフォーマットを満たしていないことです。過去に標準化されたすべての IPv6 拡張ヘッダが図 4.21 を満たしているわけではないので、RFC 6564 は「将来定義される IPv6 拡張ヘッダは」という前提つきになっています。

[†42] RFC 6564：S. Krishnan, J. Woodyatt, E. Kline, J. Hoagland, M. Bhatia, “A Uniform Format for IPv6 Extension Headers”, 2012 年 4 月

4.3.7 IPv6 拡張ヘッダが付いたパケットを破棄するルータの存在

2016 年に発行された RFC 7872[†43] は、2014 年と 2015 年に行われた調査を紹介する Informational な RFC です。RFC 7872 は、IPv6 拡張ヘッダが含まれるパケットを破棄するルータがインターネット上で実運用されていることを紹介しています。

RFC 7872 には、IPv6 拡張ヘッダが追加されたパケットのうち平均して 50% 以上が破棄されたという調査も含まれています。また、このようなパケットの破棄が、トランジットを行っている AS において発生している可能性も示唆されています。IPv6 では、フラグメンテーションにも IPv6 拡張ヘッダを利用しているため、フラグメンテーションされる IPv6 パケットもこの問題の影響を受ける可能性があります。

RFC 7872 では、このような状況は時間とともに改善されると期待しているものの、本書執筆時点では IPv6 拡張ヘッダを付けることによってパケットが途中経路で破棄される可能性が依然として考えられるでしょう。

[†43] RFC 7872 : F. Gont, J. Linkova, T. Chown, W. Liu, “Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World”, 2016 年 6 月

第5章

ICMPv6

IPv4とIPv6の大きな違いの1つに、ICMP（Internet Control Message Protocol）の役割の変化が挙げられます。もともとICMPは、プロトコル番号が1であることからわかるように、IPv4の設計当初からインターネットを構築・運用するうえで重要視されていたプロトコルです。IPv6で使われる**ICMPv6**では、その役割がさらに大きくなり、もはやIPv4で使われるICMPと似て非なるプロトコルだと考えたほうがいいでしょう。バージョン番号が4から6に変わっただけでなく、ICMPとICMPv6はまったく異なるプロトコルなのです。実際、ICMPv6には58番という、ICMPとはまったく違うプロトコル番号が割り当てられています。

ICMPの正式名称であるInternet Control Message Protocolは、そのまま日本語にすれば、「インターネットをコントロールするメッセージについてのプロトコル」となります。この名称からは、インターネットプロトコルの動作を制御するためのプロトコルという印象を強く受けるかもしれません。確かにICMPには、通信エラーの伝達のほか、Path MTU discoveryのようなインターネットプロトコルを制御するための用途があります。ただ、ネットワークの疎通確認のために利用される`ping`コマンドや、目的地までの経路確認に利用される`traceroute`コマンドなど、インターネットプロトコルを運用するエンジニアが状況把握のために使うというのがICMPの実際の主な用途だったといえます。

これに対し、ICMPv6は、文字どおり「インターネットをコントロールするメッセージ」について規定したプロトコルという色が強くなっています。たとえば、リンク層であるイーサネットアドレスをIPv4アドレスから取得するという、インターネットプロトコルの動作に不可欠な機能は、IPv4ではARP（Address Resolution Protocol）という別のプロトコルとして定義されていましたが、IPv6ではICMPv6に統合されました。具体的には、IPv6ではICMPv6メッセージを利用した近隣探索プロトコル（Neighbor Discovery Protocol）でリンク層アドレスを解決します。近隣探索プロトコルについては第6章で解説します。

また、マルチキャストを利用するための機能も、やはりIPv4では別のプロトコルだったIGMP（Internet Group Management Protocol）からICMPv6へと統合されています。ICMPv6メッセージを利用したMLD（Multicast Listener Discovery）については第11章で解説します。

表5.1に、ICMPv6で実現するインターネットプロトコルの制御機能と、そのIPv4における

(cc) BY NC SA

実現手段とを対比します。

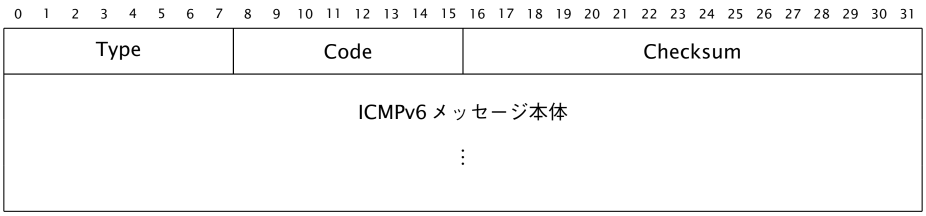
▶表5.1　さまざまな機能のICMPv6への統合

機能	IPv4における手段	IPv6における手段
エラーメッセージ、情報メッセージなど	ICMP	ICMPv6
IPアドレスからリンク層アドレスの解決	ARP	近隣探索（ICMPv6）
リンク層アドレスからIPアドレスの解決	RARP	近隣探索（ICMPv6）
マルチキャストの機能	IGMP	MLD（ICMPv6）

5.1 ICMPv6フォーマット

ICMPv6ではさまざまなメッセージが定義されており、それをIPv6パケットに格納してやり取りすることで、インターネットの制御に必要なさまざまな機能を実現します。この節では、まず各種ICMPv6メッセージの概要と、それらのフォーマットについて説明します。ICMPv6メッセージを活用する方法については、以降の各節で、さまざまな機能ごとに説明していきます。

ICMPv6は、RFC 4443[†1]で定義されています。RFC 4443では、さまざまなICMPv6メッセージの基本的なフォーマットが図5.1のように示されています。



▶図5.1　ICMPv6ヘッダの基本的なフォーマット

- Type（8ビット）

  ICMPv6メッセージの種類（ICMPv6タイプ）を示しています。

- Code（8ビット）

  ICMPv6タイプごとに定義されている値です。各ICMPv6タイプをさらに細かく分類するために用いられます。

- Checksum（16ビット）

  ICMPv6ヘッダのTypeフィールドから始まるICMPv6メッセージ全体と、その先頭に疑似IPv6ヘッダを追加したデータに対して計算したチェックサムです。5.1.1項を参照してください。

†1　RFC 4443：A. Conta, S. Deering, M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, 2006年3月

- ICMPv6 メッセージ本体

  ICMPv6 メッセージごとに異なる内容のデータです。ICMPv6 タイプに応じてフォーマットも異なります。

5.1.1 ICMPv6 でのチェックサム

ICMPv6 では、**チェックサム**の計算方法が IPv4 における ICMP とは異なります。IPv4 における ICMP では、ICMP のみを対象としてチェックサムを計算しますが、ICMPv6 では IPv6 ヘッダの一部を**疑似 IPv6 ヘッダ**（pseudo-header、図 4.12 参照）という形で含め、ICMPv6 ヘッダとともにチェックサムを計算します。なぜ疑似 IPv6 ヘッダを含めた形でチェックサムを計算するかというと、IPv6 ヘッダ自体にチェックサムのためのフィールドが存在しないからです。IPv4 では、IPv4 ヘッダ自体にチェックサムのためのフィールドがあったので、ICMP のチェックサム計算で IPv4 ヘッダの情報を気にする必要はありませんでした。

疑似 IPv6 ヘッダを含めた ICMPv6 のチェックサム計算では、RFC 8200[†2] の Section 8.1 に記載された方法が使われます。これは、4.2 節で説明している、IPv6 に対する上位層（TCP や UDP）でのチェックサム計算の方法と同じです。チェックサム計算の際、疑似 IPv6 ヘッダの Next Header の値としては、ICMPv6 を示す 58 が使われます。

5.1.2 2 種類の ICMPv6 メッセージとメッセージタイプ

ICMPv6 メッセージは、エラーメッセージと情報メッセージの 2 種類に大きく区別されています。各メッセージには、タイプを示す値が IANA によって割り当てられています[†3]。

タイプ値の最上位ビットが 0、すなわち 0 から 127 までのタイプ値はエラーメッセージに割り当てられています。表 5.2 に、ICMPv6 の主なエラーメッセージをまとめます。

▶ 表 5.2 ICMPv6 エラーメッセージ

タイプ	名前	定義されている RFC
0	Reserved	-
1	Destination Unreachable	RFC 4443
2	Packet Too Big	RFC 4443
3	Time Exceeded	RFC 4443
4	Parameter Problem	RFC 4443
100	Private Experimentation（実験用）	RFC 4443
101	Private Experimentation（実験用）	RFC 4443

タイプ値の最上位ビットが 1、すなわち 128 から 255 までのタイプ値は情報メッセージに割り当てられています。表 5.3 に、ICMPv6 の主な情報メッセージをまとめます。

†2 RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017 年 7 月
†3 “Internet Control Message Protocol version 6 (ICMPv6) Parameters”：https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml

CC BY NC SA

▶ 表 5.3　ICMPv6 情報メッセージ

タイプ	名前	定義されている RFC
128	Echo Request	RFC 4443
129	Echo Reply	RFC 4443
130	Multicast Listener Query	RFC 2710[†4]
131	Multicast Listener Report	RFC 2710
132	Multicast Listener Done	RFC 2710
133	Router Solicitation	RFC 4861[†5]
134	Router Advertisement	RFC 4861
135	Neighbor Solicitation	RFC 4861
136	Neighbor Advertisement	RFC 4861
137	Redirect	RFC 4861
143	Version 2 Multicast Listener Report	RFC 3810[†6]
151	Multicast Router Advertisement	RFC 4286[†7]
152	Multicast Router Solicitation	RFC 4286
153	Multicast Router Termination	RFC 4286
200	Private experimentation	RFC 4443
201	Private experimentation	RFC 4443

5.2 ICMPv6エラーメッセージ

2018年現在、IANAによって割り当てられているICMPv6エラーメッセージのタイプは表5.2の7項目のみです。ICMPv6エラーメッセージは、ICMPv6情報メッセージと比べると種類は少ないといえます。

ICMPv6のエラーメッセージは、そのエラーを発生させたパケット本体を、ICMPv6エラーメッセージのパケットがIPv6の最小MTUである1280オクテットを超えない範囲でデータ部分に含みます。ICMPv6エラーメッセージに含まれるデータ部分から、そのメッセージが渡される上位プロトコルが判断されます。たとえば、ICMPv6エラーメッセージのデータ部分の先頭がTCPヘッダであれば、そのICMPv6メッセージはTCPを処理するプロトコルスタックに手渡されます。

ICMPv6エラーメッセージが発生させるトラフィックが他のトラフィックを圧迫してしまうことを防ぐために、IPv6対応しているノードは、ICMPv6エラーメッセージを送信するレートに制限をかけることが求められています。たとえば、`traceroute`コマンドはDestination UnreachableやTime Exceededを利用していますが、必ずしもIPv6ノードがICMPv6エラーメッセージを返してくれるとは限りません。

†4　RFC 2710：S. Deering, W. Fenner, B. Haberman, “Multicast Listener Discovery (MLD) for IPv6”, 1999年10月

†5　RFC 4861：T. Narten, E. Nordmark, W. Simpson, H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, 2007年9月

†6　RFC 3810：R. Vida, L. Costa, “Multicast Listener Discovery Version 2 (MLDv2) for IPv6”, 2004年6月

†7　RFC 4286：B. Haberman, J. Martin, “Multicast Router Discovery”, 2005年12月

ICMPv6 エラーメッセージに対して ICMPv6 エラーメッセージを返すことは禁止されています。

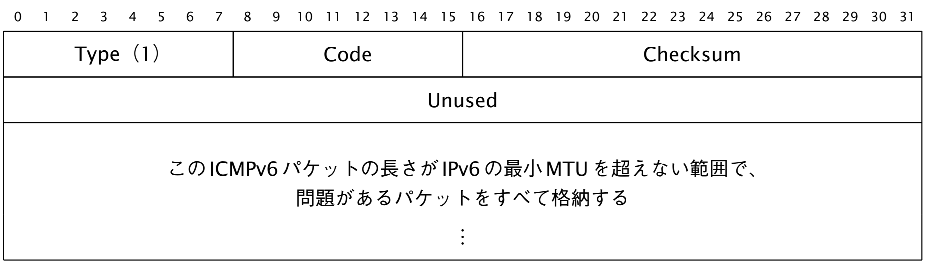
IPv6 マルチキャストアドレスを宛先とするパケットに対して、ICMPv6 エラーメッセージを送信することも禁止されています。ただし、IPv6 マルチキャストで Path MTU discovery を行うための Packet Too Big と、Parameter Problem メッセージの Code 2 かつ Option Type の上位 2 ビットが 10 の 2 つが例外として RFC 4443 に記載されています。

以下、RFC 4443 で定義されている ICMPv6 エラーメッセージをそれぞれ解説します。

5.2.1 Destination Unreachable メッセージ

宛先までパケットを送信できない場合に使われるのが Destination Unreachable メッセージです。Destination Unreachable メッセージを送信するのは、ルータもしくはパケット生成ノードのネットワーク層（IPv6 のレイヤー）です。ただし、輻輳によって破棄されたパケットを原因とした ICMPv6 メッセージの生成は禁止されています。

図 5.2 に Destination Unreachable メッセージのフォーマットを示します。



▶ 図 5.2　Destination Unreachable メッセージフォーマット

- Type（8 ビット）

  Destination Unreachable メッセージの ICMPv6 タイプを示す 1 が入ります。

- Code（8 ビット）

  宛先までパケットを送信できなかった原因を表す数値です。以下のような値が定義されています。

値	意味	説明
0	No route to destination	宛先までの経路が存在しなかった
1	Communication with destination administratively prohibited	宛先との通信が設定によって禁止されている
2	Beyond scope of source address	送信元アドレスのスコープ外の宛先である
3	Address unreachable	アドレスが到達不能である
4	Port unreachable	ポートが到達不能である
5	Source address failed ingress/egress policy	内部向けもしくは外部向けのポリシーによって許可されない送信元アドレスだった
6	Reject route to destination	宛先に対する経路が却下された

ノードのルーティングテーブル内に適合する経路が存在しないことが原因でパケットが送信できなかった場合、Codeフィールドの値には0を設定します。この値は、ノード内にデフォルトルートが存在しない場合にのみ発生します。なぜなら、デフォルトルートが存在する場合や、ルーティングテーブル内に明示的な経路が存在しない場合には、デフォルトルートが選択されるためです。

ファイアウォールの設定など、運用上の理由でパケットが送信できなかった場合、Codeフィールドの値には1を設定します。

宛先が送信元アドレスのスコープ外である場合には、Codeフィールドの値を2に設定します。この状況が発生するのは、送信元アドレスのスコープが宛先アドレスのスコープよりも小さい場合だけです。たとえば、送信元アドレスとしてリンクローカルユニキャストアドレスが使用されたパケットで、宛先として指定されたアドレスのスコープがグローバルであり、かつ、そのグローバルな宛先へのインターフェースと受信インターフェースが異なるリンクに属している場合には、Codeフィールドの値を2に指定します[†8]。

該当する他のCodeが存在しない場合には、Codeフィールドの値を3に設定します。たとえば、パケットを送出するリンク層で発生する問題などが挙げられます。

宛先アドレスで、トランスポート層プロトコルが該当するポート番号で待ち受けを行っていない場合には、Codeフィールドの値を4に設定します。たとえば、UDPやTCPで該当するポート番号でリスンしているプロセスが存在しないといった状況では、このコード番号を使います。

内部向けもしくは外部向けのフィルタリングポリシーによって許可されないソースアドレスだった場合には、Codeフィールドの値を5に設定します。1よりもさらに詳細な情報を伝えるときに使う値です。

宛先に対する経路が却下されるときには、Codeフィールドの値を6に設定します。たとえば、ルータにおいて特定のプレフィックス向けのトラフィックをすべて拒否するように設定されている場合に、このコードを利用します。1よりもさらに詳細な情報を伝えるときに使う値です。

- Checksum（16ビット）

  ICMPv6ヘッダのTypeフィールドから始まるICMPv6メッセージ全体と、その先頭に疑似IPv6ヘッダを追加したデータに対して計算したチェックサムです。5.1.1項を参照してください。

- Unused（32ビット）

  未使用のフィールドです。ここには0が入り、受信側では無視されます。

ICMPv6メッセージを受け取ったIPv6の層では、上位層にそれを通知する必要があります。たとえば、TCPやUDPのようなトンランスポート層プロトコルでは、Destination Unreachableメッセージを利用しています。

†8　同一リンク内であれば、リンクローカルユニキャストアドレスからグローバルユニキャストアドレスへの通信も可能です。同一リンク内で異なるプレフィックス同士が通信する場合に関しては6.6節を参照してください。

CC BY NC SA

NOTE

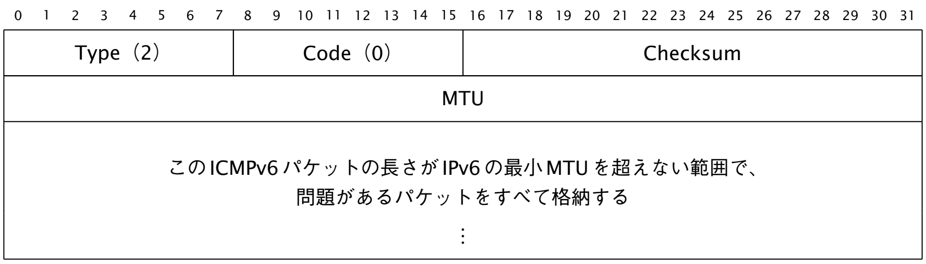
Destination Unreachable メッセージは、セキュリティ上の理由で無効化されることもあります。

5.2.2 Packet Too Big メッセージ

Packet Too Big メッセージは、ルータが送信します。Packet Too Big メッセージは、ルータが転送するパケットサイズが、そのルータの出力側ネットワークインターフェースのMTUよりも大きいためにパケットが転送できないことを、ルータが送信者に知らせるときに利用されます。

Packet Too Big メッセージは、他のICMPv6エラーメッセージとは異なり、IPv6マルチキャストを宛先とするパケットに対する応答として送られることもあります。リンク層においてブロードキャストアドレスおよびマルチキャストアドレスで送信されているパケットに対する応答としてICMPv6 Packet Too Big メッセージが送られることもあります。これらは、IPv6マルチキャストにおけるPath MTU discoveryを実現するためです。Path MTU discoveryについては第10章で解説します。

図5.3にPacket Too Big メッセージのフォーマットを示します。



▶ 図5.3　Packet Too Big メッセージフォーマット

- Type（8ビット）

  Packet Too Big メッセージのICMPv6タイプを示す2が入ります。

- Code（8ビット）

  0が入ります。受信側で無視されます。

- Checksum（16ビット）

  ICMPv6ヘッダのTypeフィールドから始まるICMPv6メッセージ全体と、その先頭に疑似IPv6ヘッダを追加したデータに対して計算したチェックサムです。5.1.1項を参照してください。

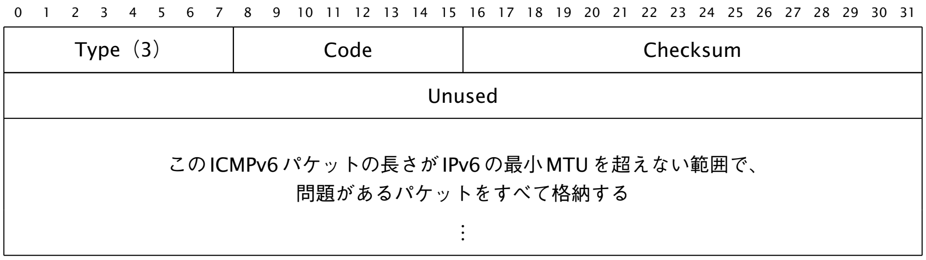
- MTU（32ビット）

  Packet Too Big メッセージを発生させる原因となった次ホップについて、リンク層のMTUの値を含めるフィールドです。

5.2.3 Time Exceeded メッセージ

Time Exceeded メッセージは、その名のとおり、有効期間が過ぎてしまったことを示すための ICMPv6 メッセージです。

図 5.4 に Time Exceeded メッセージのフォーマットを示します。



▶ 図 5.4　Time Exceeded メッセージフォーマット

- Type（8 ビット）

  Time Exceeded メッセージの ICMPv6 タイプを示す 3 が入ります。

- Code（8 ビット）

  どのような状況で有効期間が切れたのかを示します。以下の 2 つの値が定義されています。

値	意味	説明
0	Hop limit exceeded in transit	転送の際にホップリミットを超えた
1	Fragment reassembly time exceeded	フラグメントの再構成時間を過ぎた

  ルータは、受け取ったパケットの Hop Limit が 0 であるか、転送に際して 1 減算した結果として Hop Limit の値が 0 になった場合は、そのパケットを破棄すると同時に、「Code フィールドの値を 0 に設定した Time Exceeded メッセージ」をパケットに記載された送信元アドレス宛に送信します。通信経路を調べるときなどに使われる `traceroute` コマンドでは、この仕様を利用しています。

  ノードは、最初のフラグメントを受け取ってから 60 秒以内にパケットの再構成に必要なすべてのフラグメントを受け取れない場合には、「Code フィールドの値を 1 に設定した Time Exceeded メッセージ」をパケットに記載された送信元アドレス宛に送信します。これは、RFC 8200 の Section 4.5 で規定されている仕様です。

- Checksum（16 ビット）

  ICMPv6 ヘッダの Type フィールドから始まる ICMPv6 メッセージ全体と、その先頭に疑似 IPv6 ヘッダを追加したデータに対して計算したチェックサムです。5.1.1 項を参照してください。

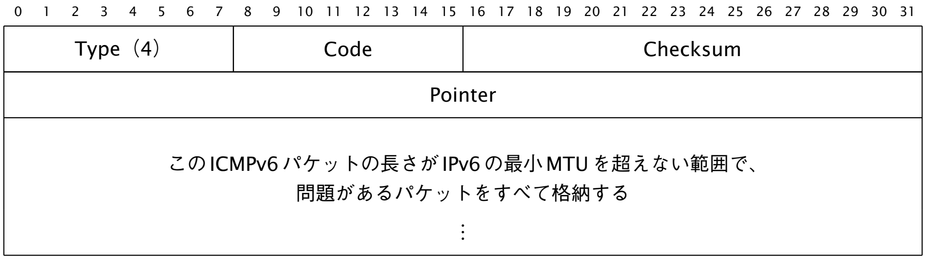
- Unused（32 ビット）

  未使用のフィールドです。ここには 0 が入り、受信側では無視されます。

5.2.4 Parameter Problem メッセージ

IPv6 ノードがパケットを処理中に IPv6 ヘッダもしくは IPv6 拡張ヘッダに問題を発見した場合、そのパケットを破棄する必要があります。同時に、Parameter Problem メッセージをパケット送信者に送ることが推奨されています。Parameter Problem メッセージは、問題が発生したこと伝えると同時に、問題を発生させた箇所を伝えるためのメッセージだといえます。

図 5.5 に Parameter Problem メッセージのフォーマットを示します。



▶ 図 5.5 Parameter Problem メッセージフォーマット

- Type（8 ビット）

  Parameter Problem メッセージの ICMPv6 タイプを示す 4 が入ります。

- Code（8 ビット）

  どのような問題が起きたのかを示します。以下の 3 つの値が定義されています。

値	意味	説明
0	Erroneous header field encountered	誤ったヘッダフィールドに遭遇した
1	Unrecognized Next Header type encountered	未知の Next Header タイプに遭遇した
2	Unrecognized IPv6 option encountered	未知の IPv6 オプションに遭遇した

  Code 1 と Code 2 は、Code 0 をさらに詳細に分類したものです。

- Checksum（16 ビット）

  ICMPv6 ヘッダの Type フィールドから始まる ICMPv6 メッセージ全体と、その先頭に疑似 IPv6 ヘッダを追加したデータに対して計算したチェックサムです。5.1.1 項を参照してください。

- Pointer（32 ビット）

  Pointer フィールドでは、問題が発生した箇所を、IPv6 ヘッダの先頭からのオフセットとして示します。たとえば、Code フィールドの値が 1 で、Pointer フィールドの値が 40

の場合、IPv6ヘッダ（40オクテット）の直後に続くIPv6拡張ヘッダのNext Headerの値に問題があることがわかります。

エラーを発生させたパケット本体がICMPv6メッセージの最大値を超え、ICMPv6メッセージのデータ部分に収まりきらない大きさだった場合、Pointerフィールドの示す値がICMPv6パケットの末尾を越える場合もあります。

5.3 ICMPv6情報メッセージ

ICMPv6情報メッセージのタイプは、ICMPv6エラーメッセージタイプと比べると、たくさんの種類が定義されています。ただし、ICMPv6エラーメッセージについてはIANAで割り当てられているものがすべてRFC 4443で定義されているのに対し、ICMPv6情報メッセージの多くは他のさまざまなRFCで定義されているものです（118ページの表5.3を参照）。

ここでは、RFC 4443で定義されている2つのICMPv6情報メッセージであるICMPv6 Echo RequestメッセージとEcho Replyメッセージについてだけフォーマットを紹介します。マルチキャストに関連するICMPv6情報メッセージは11.5.1項で、近隣探索プロトコルに関連するICMPv6情報メッセージについては第6章で解説します。

NOTE

受信側ノードは、未知のタイプのICMPv6情報メッセージを受け取った場合、そのICMPv6情報メッセージを破棄します。

5.3.1 ICMPv6 Echo RequestメッセージとEcho Replyメッセージ

IPv4のICMPと同様に、ICMPv6にもEcho RequestメッセージとEcho Replyメッセージがあります。Echo RequestメッセージとEcho Replyメッセージは、ノードの生存確認やネットワークの疎通確認などで利用されます。

Echo Requestメッセージは、Echo Replyメッセージを送信することを要求するメッセージです。Echo Requestメッセージに対する応答としてEcho Replyメッセージが返信されます。これらのメッセージを利用する代表的なコマンドとしてpingコマンドが挙げられます。

すべてのノードは、IPv6対応にあたり、Echo RequestメッセージとEcho Replyメッセージを処理するための機能を実装する必要があります。それと同時に、Echo Requestメッセージを送信してEcho Replyメッセージを受信する、アプリケーション層のインターフェースを実装することが推奨されています。

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Type（128）	Code（0）	Checksum
Identifier		Sequence Number
Data ・・・		

▶図5.6　Echo Requestメッセージ

0–7	8–15	16–31
Type（129）	Code（0）	Checksum
Identifier		Sequence Number
Data ...		

▶ 図 5.7 Echo Reply メッセージ

- Type（8 ビット）

  Echo Request メッセージの場合は 128、Echo Reply メッセージの場合は 129 を指定します。

- Code（8 ビット）

  Code フィールドの値は、Echo Request メッセージも Echo Reply メッセージも常に 0 であり、受信側では無視されます。

- Checksum（16 ビット）

  ICMPv6 ヘッダの Type フィールドから始まる ICMPv6 メッセージ全体と、その先頭に疑似 IPv6 ヘッダを追加したデータに対して計算したチェックサムです。5.1.1 項を参照してください。

- Identifier（16 ビット）

  Echo Reply メッセージと Echo Request メッセージの対応関係を明示するために使う識別子です。0 でもかまいません。

- Sequence Number（16 ビット）

  Echo Reply メッセージと Echo Request メッセージの対応関係を明示するために使うシーケンス番号です。0 でもかまいません。

- Data（可変長）

  Echo Request メッセージには任意のデータを含めることができます。対応する Echo Reply メッセージでは、そのデータをそのままコピーして返します。

Echo Reply メッセージでは、Echo Request メッセージに記載されている識別子、シーケンス番号、データをそのまま返します。Echo Request メッセージの送信側は、受け取った Echo Reply メッセージに記載されている情報を確認することで、どの Echo Request メッセージが宛先ノードに到達したのかを確認できるという仕組みです。

BY NC SA

[書籍版では空白のページです]

BY NC SA

第6章

近隣探索プロトコル

ICMPv6を用いて実現される**近隣探索プロトコル**（Neighbor Discovery Protocol、**NDP**）は、IPv6の根幹となる非常に大きな特徴であり、IPv4でのARP（Address Resolution Protocol）、ICMP Router Discovery、ICMP Redirectで実現されていた内容が含まれています。また、IPv4には存在しなかった機能として、近隣ノードが到達不能であることを検知できる近隣不到達性検知（Neighbor Unreachability Detection、NUD）もあります。さらに、ICMPv6メッセージを組み合わせて自動的にIPv6アドレスを生成するSLAAC（Stateless Address Autoconfiguration）も用意されています。

IPv6の近隣探索プロトコルは、IPv4でリンク層アドレス解決に使われるARPを引き合いに出して説明されることがありますが、ARPと比べると以下のような点が改善されています（RFC 4861のSection 3.1も参照してください）。

- アドレス解決と同時にルーティングに必要な情報も解決できる
- リンク層アドレス解決のためのマルチキャストを分散できる
- Redirectメッセージが近隣探索プロトコルに含まれているので、別途アドレス解決をする必要がない
- 近隣不到達性検知がある
- サイトリナンバリングが容易になる可能性
- リンクの外部からリダイレクトの情報やルータの情報がくることがない（Hop limit 255の効能）
- ARPに依存せず、ネットワーク層の認証やセキュリティ機能を使えるようになる

近隣探索プロトコルそのものは、RFC 4861[†1]で定義されています。RFC 4861で示されているのは、近隣探索プロトコルで利用されるICMPv6メッセージのフォーマットや、各メッセージの送受信に伴う処理が中心です。SLAACについては、別のRFC 4862[†2]でまとめられてい

†1 RFC 4861：T. Narten, E. Nordmark, W. Simpson, H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, 2007年9月

†2 RFC 4862：S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration”, 2007年9月

CC BY NC SA

ます。

本章では、RFC 4861 の内容を中心に、近隣探索プロトコルについて基本的な内容をまとめます。SLAAC については第 7 章で扱います。

6.1 近隣探索プロトコルの機能と利用するメッセージ

RFC 4861 では、近隣探索プロトコルが提供する機能を以下のように要約しています。

- リンク上のルータを探す機能（Router Discovery）
- ルータを経由せずに到達できる IPv6 アドレスの範囲を知る機能（Prefix Discovery）
- リンクの MTU などの情報を知る機能（Parameter Discovery）
- インターフェースに対してステートレスにアドレスを振る機能（Address Autoconfiguration）
- IPv6 アドレスからリンク層のアドレスを解決する機能（Address Resolution）
- 宛先アドレスをもとに、次にパケットを送出すべき IPv6 アドレスを知る機能（Next-hop Determination）
- 近隣ノードに到達できなくなったことを知る近隣不到達性検知（Neighbor Unreachability Detection）
- 利用するアドレスが他のノードで使われていないかを確認する機能（Duplicate Address Detection）
- ルータからホスト[†3]へ、より適切な送出先を伝える方法（Redirect）

上記の機能を実現するために、近隣探索プロトコルでは、次の 5 つの ICMPv6 メッセージを組み合わせて利用します。それぞれの ICMPv6 メッセージが運ぶオプションデータによって、さまざまな機能を実現しているのです。これらの ICMPv6 メッセージは近隣探索メッセージと呼ばれ、RFC 4861 で定義されています。

- Router Advertisement メッセージ（6.2.1 項）
- Router Solicitation メッセージ（6.2.2 項）
- Neighbor Solicitation メッセージ（6.3.1 項）
- Neighbor Advertisement メッセージ（6.3.2 項）
- Redirect メッセージ（6.4 節）

Router Advertisement の Advertisement は、「広告」という意味を持つ英単語です。Router Advertisement（ルータ広告、RA）メッセージは、ルータの存在をサブネット内に広告すると同時に、ルータと通信する際に必要となる各種情報を伝える役割があります。ルータは、Router Advertisement メッセージを定期的にリンク内にマルチキャストすることによって、サブネット内のノードに存在を通知できます。また、後述する Router Solicitation に対する返答として送信することも可能です。

[†3] IPv6 の基本仕様を示した RFC 8200 での定義では、「ホスト」はルータではないノード、と定義されています。1.1.3 項も参照。

CC BY NC SA

Router SolicitationのSolicitationは、「要請」や「懇願」、「懇請」といった意味を持つ英単語です。Router Solicitation（ルータ要請、RS）メッセージは、Router Advertisementメッセージの送信をただちに行うように要求するためのメッセージです。

Neighbor Solicitation（近隣要請、NS）メッセージは、同一サブネット内に接続している近隣ノードのリンク層アドレスを得るために送信されるメッセージです。近隣ノードとの通信が不可能になっていないかどうか（近隣不到達性）を検知したり、同一のIPv6アドレスを利用している他ノードが存在しているかどうかを確認するDAD（Duplicate Address Detection、重複アドレス検知）で使われたりします。

Neighbor Advertisement（近隣広告、NA）メッセージは、Neighbor Solicitationメッセージに対する返答として送信できるメッセージです。自分のリンク層アドレスが変わったことを通知するために、Neighbor Advertisementメッセージを自発的に送信することも可能です。

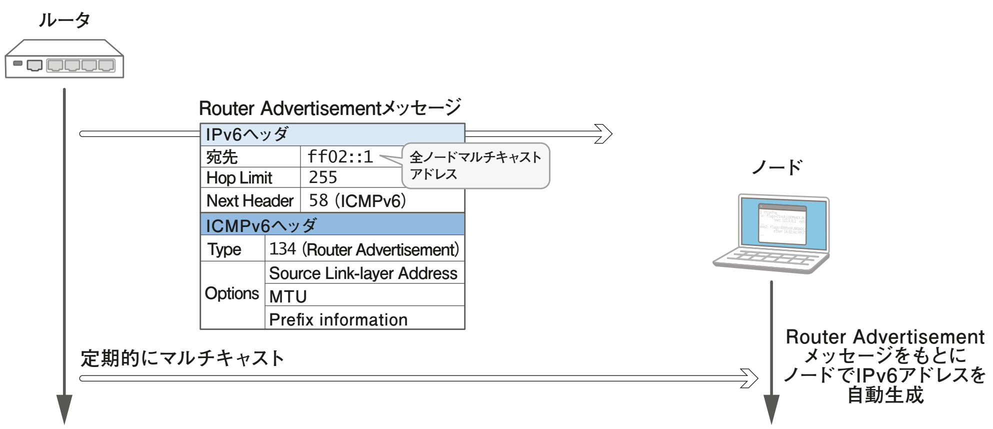
Redirectメッセージはルータがホストに対して、宛先に対するより最適な次ホップノードを伝えるためのメッセージです。

これらの近隣探索メッセージを送信するときは、IPv6ヘッダのHop Limitフィールドに255という値を設定します。これにより受信側では、Hop Limitが255ではないものを受け取ったとき、そのICMPv6メッセージがルータを越えたことを検出できます。

6.2 ルータとプレフィックス情報の発見

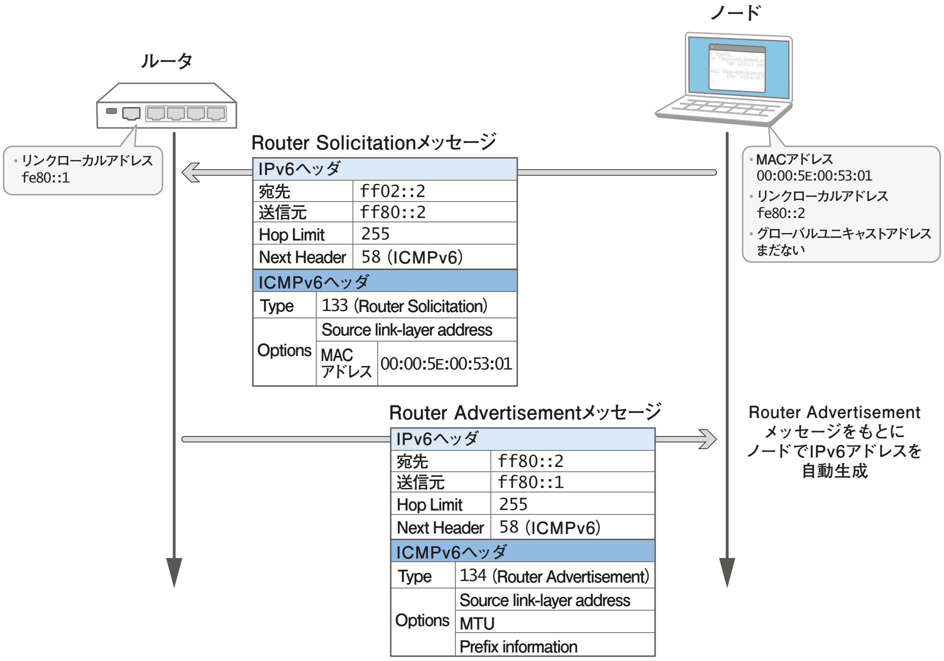
近隣探索プロトコルでホストがルータを発見したり、ルーティング設定に必要なプレフィックス情報をホストが知るためには、ICMPv6のRouter Advertisementメッセージを利用します。「Router Advertisement」を日本語に直訳すると「ルータの広報」という意味になることからわかるように、Router Advertisementメッセージはルータからホストに対して送信されます。

ルータでは、ホストにおけるネットワーク設定に必要な情報を含むRouter Advertisementを、定期的にマルチキャストで送信します。これにより、そのルータのサブネットに接続している機器は、そのサブネットを利用した通信を行うために必要な情報を得られます（図6.1）。



▶ 図6.1 Router Advertisementメッセージをマルチキャスト

ルータから定期的にRouter Advertisementメッセージをマルチキャストするだけでなく、ネットワークに接続された各機器の側からルータに対してRouter Advertisementメッセージの送信を要求することも可能です。このときに使われるのがRouter Solicitationメッセージです（図6.2）。



▶図6.2　Router Solicitationメッセージへの返答としてのRouter Advertisementメッセージ

Router Advertisementメッセージを受け取った各機器では、そこに含まれているプレフィックス情報にインターフェース識別子を加えることで、各自のIPv6アドレスを自動生成します。このIPv6アドレスの自動設定（SLAAC）に関しては、第7章を参照してください。

以下、ICMPv6のRouter Advertisementメッセージと Router Solicitationメッセージのフォーマットを説明します。さらに、近隣探索プロトコルにおけるルータとプレフィックス情報の発見ではルータとホストの役割や挙動が異なるので、ルータ側で行う処理とホスト側で行う処理についてそれぞれ解説します。

6.2.1　Router Advertisementメッセージ

Router Advertisementは、タイプ134のICMPv6メッセージです。Router Advertisementメッセージのフォーマットは RFC 4861に記載されています。Router Advertisementは ICMPv6の一部ですが、ICMPv6の基本仕様であるRFC 4443[†4]ではなく、近隣探索の仕様を示したRFC 4861という別のRFCで定義されているのです。

Router Advertisementメッセージを運ぶIPv6パケットのIPv6ヘッダでは、送信元IPv6アドレスに、送信に使われるネットワークインターフェースに設定されたリンクローカルユニ

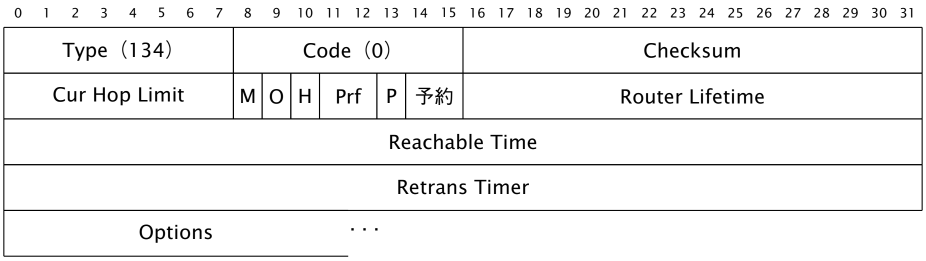
†4　RFC 4443：A. Conta, S. Deering, M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, 2006年3月

キャストアドレスを指定する必要があります。

Router Advertisementメッセージを運ぶIPv6パケットでは、IPv6ヘッダの宛先IPv6アドレスとして、通常は全ノードアドレス（`ff02::1`）を指定します。Router Solicitationメッセージに対する返答の場合は、Router Solicitationメッセージに記載された送信元IPv6アドレスを指定します。

前述したように、Router Advertisementメッセージを運ぶIPv6パケットでは、IPv6ヘッダのHop Limitの値を255にします。これにより、他のリンクに接続された機器から誤ってRouter Advertisementメッセージが転送されてきても、その内容に影響されないようにできます。

図6.3にRouter Advertisementメッセージのフォーマットを示します。



▶図6.3　Router Advertisementメッセージフォーマット

- Type（8ビット）

  Router Advertisementメッセージであることを示す134が入ります。

- Code（8ビット）

  0が入ります。

- Checksum（16ビット）

  ICMPチェックサムを設定します。

- Cur Hop Limit（8ビット）

  そのルータを通じて転送されていくIPv6パケットのHop Limitフィールドについて、ルータが推奨する値を示します。値は8ビットの符号なし整数（unsigned integer）で指定します。ルータによって特に推奨値を指定しない場合は、0にします。

- M（1ビット）

  「Managed address configuration」を意味するフラグです。DHCPv6によるIPv6アドレス設定が利用可能であることを示すためのフラグです。DHCPv6によるIPv6アドレス設定が利用可能である場合に、このフラグが1に設定されます。このフラグは、あくまでDHCPv6が利用可能であることを示すだけであり、このフラグが0である場合にDHCPv6を利用することが禁止されているわけではありません。

- O（1ビット）

  「Other configuration」を示すフラグです。IPv6アドレス以外の情報をDHCPv6で取得

することが可能である場合は、このフラグを1に設定します。たとえば、DNSに関連する情報がDHCPv6で提供されている場合には、このフラグを設定します。OフラグもMフラグ同様に、利用可能であることを示すものであり、このフラグが0という状況においてステートレスなDHCPv6の利用が禁止されているわけではありません。

- H（1ビット）
  「Mobile IPv6 Home Agent」を示すフラグです。RFC 3775[†5]で定義されています。
- Prf（2ビット）
  「Default Router Preference」を示すフラグです。RFC 4191[†6]で定義されています。
- P（1ビット）
  「Neighbor Discovery Proxy」を示すフラグです。RFC 4389[†7]で定義されています。
- 予約（2ビット）
  未使用領域です。このフィールドの値は0であることが求められます。
- Router Lifetime（16ビット）
  自分がデフォルトルータとして有効な期間をホストに伝えるためのフィールドです。16ビットの符号なしの整数により、秒単位で指定します。このフィールドの値が0である場合は、自身をデフォルトルータとして利用してはならないことをホストに対して示します。16ビットの符号なしの整数なので、フィールドとしては65535までの値を表現可能ですが、RFC 4861ではこのフィールドで利用可能な最大値を9000秒にするという制約があります。ただし、RFC 4861を更新するRFC 8319で9000秒という制約は撤廃され、最大値が65535となっています。
- Reachable Time（32ビット）
  近隣不到達性検知アルゴリズムで使われるフィールドです。32ビットの符号なしの整数です。単位はミリ秒です。0は、このルータでは未定義という意味を持ちます。
- Retrans Timer（32ビット）
  ホストに対してNeighbor Solicitationメッセージの再送信まで待つべき時間を指定するフィールドです。ミリ秒単位で32ビットの符号なしの整数を指定します。0は、そのルータでは未定義という意味を持ちます。
- Options（可変長）
  近隣探索メッセージのためのオプション（6.5節で説明します）を含めるフィールドです。Router Advertisementメッセージに含められるのは、RFC 4861で定義されているSource Link-layer Addressオプション（6.5.2項）、MTUオプション（6.5.5項）、Prefix

---

†5　RFC 3775：D. Johnson, C. Perkins, J. Arkko, “Mobility Support in IPv6”, 2004年6月
†6　RFC 4191：R. Draves, D. Thaler, “Default Router Preferences and More-Specific Routes”, 2005年11月
†7　RFC 4389：D. Thaler, M. Talwar, C. Patel, “Neighbor Discovery Proxies (ND Proxy)”, 2006年4月

CC BY NC SA

Information オプション（6.5.3 項）のほか、RFC 8106[†8]で定義されている RDNSS オプション（6.5.6 項）、DNSSL オプション（6.5.7 項）などです。

RFC 4861 では、Router Advertisement メッセージに含まれるフラグは M と O のみであり、残りの 6 ビットが予約（Reserved）と定義されています。そして、その予約フィールド内の値として 0 を設定するように求めています。しかし、RFC 3775 で H フラグが定義され、また RFC 4191 で Prf として 2 ビットが使われることになっており、さらに RFC 4389 では P フラグが定義されています。そのため、実際は残り 6 ビットではなく 2 ビットだけが予約フィールドとなります。

Router Advertisement メッセージに含まれるフラグについては、上記の状況を整理し、さらに必要な場合にフラグを拡張できるように、Flags Expansion というオプションを定義した RFC 5175[†9]が発行されています。

6.2.2　Router Solicitation メッセージ

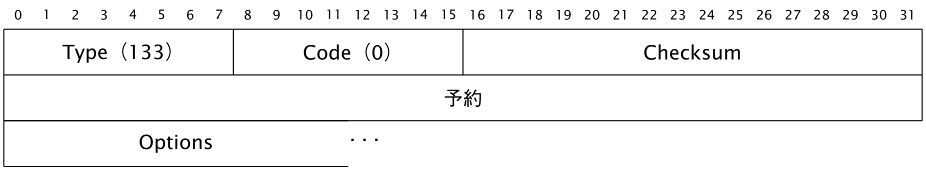
Router Solicitation は、タイプ 133 の ICMPv6 メッセージです。ルータからの Router Advertisement はマルチキャストで定期的に送信されますが、IPv6 ノード側からルータに対して Router Advertisement をただちに送信するように要求するためには、この Router Solicitation メッセージを使います。

Router Solicitation メッセージを運ぶ IPv6 パケットの IPv6 ヘッダでは、送信元 IPv6 アドレスとして、送信に使われるネットワークインターフェースに設定された IPv6 アドレスが使われます。ただし、IPv6 アドレスの自動設定を行うときなど、まだ IPv6 アドレスが設定されていない場合には、ネットワークインターフェースに IPv6 アドレスが設定されていません。その場合には、送信元 IPv6 アドレスとして、未定義アドレス（`::/128`）を使うことになっています。

Router Solicitation メッセージを運ぶ IPv6 パケットでは、IPv6 ヘッダの宛先 IPv6 アドレスとして、通常は全ルータアドレス（`ff02::2`）が使われます。

Router Solicitation メッセージを運ぶ IPv6 パケットでは、Router Advertisement メッセージと同様に、IPv6 ヘッダの Hop Limit の値を 255 とします。

図 6.4 に Router Solicitation メッセージのフォーマットを示します。



▶ 図 6.4　Router Solicitation メッセージフォーマット

---

†8　RFC 8106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2017 年 3 月

†9　RFC 5175：B. Haberman, R. Hinden, “IPv6 Router Advertisement Flags Option”, 2008 年 3 月

- Type（8ビット）

  Router Solicitationメッセージであることを示す133が入ります。

- Code（8ビット）

  0が入ります。

- Checksum（16ビット）

  ICMPチェックサムです。

- 予約（32ビット）

  未使用領域です。このフィールドの値は0であることが求められます。受信側は、このフィールドの内容を無視する必要があります。

- Options（可変長）

  近隣探索メッセージのためのオプション（6.5節で説明します）を含めるフィールドです。Router Solicitationメッセージに含められるのは、RFC 4861で定義されているSource Link-layer Addressオプション（6.5.2項）です。なお、送信元アドレスが未定義アドレスの場合は、オプションを含めることが禁止されています。

6.2.3　ルータ側の処理

ルータは、ホスト側からのRouter Solicitationメッセージによる要求に応える形で、Router Advertisementメッセージを送信します。また、それ以外に、定期的にRouter Advertisementメッセージをマルチキャストで送信します。

ルータがRouter Advertisementメッセージで送信するのは、リンク上のホストが自身のアドレスを設定するのに必要な情報です。たとえば、ルータとしてパケットを転送するネットワークプレフィックス情報や、デフォルトルータであるかどうかといった情報などをホストに対して送信します。

ネットワークプレフィックス情報については、Prefix Informationオプション（6.5.3項）で指定します。ルータは、自身が接続しているすべての外部リンクについて、ネットワークプレフィックス情報を含めるべきとされています。

デフォルトルータであるかどうかは、Router AdvertisementメッセージのRouter Lifetimeフィールドを使って示します。Router Lifetimeフィールドの値を0に設定することで、自分自身をデフォルトルータとせずにRouter Advertisementメッセージを送れます。たとえば、ホストがSLAACでIPv6アドレスを自動設定できるように必要な情報を送信しつつ、ルータとしてパケットの転送はしないという場合、このような方法が使えます。

ルータでは、Router Advertisementメッセージを送信するかどうかを、各ネットワークインターフェースごとに設定できます。Router Advertisementメッセージを送信するネットワークインターフェースについては、IPv6の全ルータアドレスのマルチキャストグループに参加する必要があります。Router Advertisementメッセージを送信しているインターフェースで、システムをシャットダウンするなどの理由により送信をやめる場合は、Router Lifetimeをゼロに設定したRouter Advertisementメッセージを送信すべきとされています。

Router Advertisementメッセージに必要なオプションは、すべて1つのRouter Advertisementメッセージに含めることが推奨されています。しかし、リンクのMTUを超えてしまう場合には、オプションを部分的に含んだ複数のRouter Advertisementメッセージに分けて送信することも可能です。ただし、一番最初にRouter Advertisementメッセージを送信するときや、Router Solicitationメッセージへの応答として送信する場合には、必要なすべてのオプションを含むことが推奨されています。

■ 定期的なRouter Advertisementメッセージの送信間隔

定期的なRouter Advertisementメッセージは、厳密に同じ時間間隔で送信されるわけではありません。送信間隔を厳密に一定にしてしまうと、複数のルータが同じタイミングでリンク上にRouter Advertisementメッセージを送信した場合に、輻輳が発生するおそれがあります。そのためRFC 4861では、Router Advertisementメッセージの送信間隔を、ルータで設定された最小時間と最大時間の間のランダムな値としています†10。

■ ルータのリンクローカルアドレスが変わる場合

近隣探索プロトコルでは、ルータのリンクローカルアドレスを利用して、そのリンク内でルータを識別します。そのため、ルータのリンクローカルアドレスを変更すべきではありません。

ただし、リンクローカルアドレスの変更が禁止されているわけではありません。RFC 4861では、非推奨としつつも、変更を行う場合に推奨される挙動が示されています。ルータのリンクローカルアドレスが変わる場合には、ただちにその変更を通知するために、古いリンクローカルアドレスでRouter Lifetimeをゼロに設定したRouter Advertisementメッセージをいくつか送信し、そのうえで、新しいリンクローカルアドレスを使ったRouter Advertisementメッセージを送信することが推奨されています。

6.2.4 ホスト側の処理

ホスト側の処理は、主に自分自身のアドレスに関する設定を行うためにルータから情報を受け取ることです。ルータに対して情報を要求するRouter Solicitationメッセージを送信することも可能です。

一方、ルータではないホストがRouter Advertisementメッセージを送信することは禁止されています。また、ホストがRouter Solicitationメッセージを受け取った場合には、何もせずに破棄すべきであるとRFC 4861に書かれています。

ホストが受け取るRouter Advertisementメッセージには、そのメッセージを送信したルータがデフォルトルータとして振る舞う期間がRouter Lifetimeフィールドで示されています。ホストは、ルータがデフォルトルータとしての有効期限が過ぎる前に新たなRouter Advertisementメッセージによって有効期限を更新しない場合、自身のDefault Router Listからそのルータに関するエントリを削除する必要があります。

---

†10 それぞれRFC 4861では`MinRtrAdvInterval`および`MaxRtrAdvInterval`として導出方法が説明されていますが、実装によってこれらのパラメータの名前は異なります。

CC BY NC SA

ホストは、近隣探索プロトコルを使うために、マルチキャストが利用可能なすべてのインターフェースで全ノードアドレスに参加する必要があります。

6.3 リンク層アドレスの解決と近隣不到達性の検知

IPv6では、IPv4でARP（Address Resolution Protocol）によって実現していたリンク層のアドレス解決を、ICMPv6を使った近隣探索プロトコルが実現しています。さらに、リンク層アドレスの解決だけではなく、ノードに到達できないこと（近隣不到達性）を検知するための仕組みも定義されています。これら、リンク層アドレスの解決と近隣不到達性の検知には、Neighbor SolicitationメッセージとNeighbor Advertisementメッセージを利用します。

以下、Neighbor SolicitationメッセージとNeighbor Advertisementメッセージのフォーマットを説明します。さらに、両メッセージの送受信に関して説明します。最後に近隣不到達性検知について解説します。

NOTE

Neighbor SolicitationメッセージとNeighbor Advertisementメッセージは、SLAACで定義されているDAD（Duplicate Address Detection、重複アドレス検知）でも利用されます。DADに関しては、7.6節で説明します。

6.3.1 Neighbor Solicitationメッセージ

IPv6ノードは、ICMPv6のNeighbor Solicitationメッセージを利用して、他のIPv6ノードのリンク層のアドレスを要求すると同時に、自分のリンク層アドレスを伝えます。

Neighbor Solicitationメッセージを送信するIPv6パケットのヘッダに記載される送信元IPv6アドレスは、ネットワークインターフェースに設定されているIPv6アドレスです。ただし、7.6節で説明するDADの間は、送信元IPv6アドレスを未定義アドレス（`::/128`）とします。

Neighbor Solicitationメッセージは、リンク層アドレスの解決を行う場合にはマルチキャストで送信されます。その際には、IPv6ヘッダの宛先IPv6アドレスとして、Target Addressフィールドに指定する値に対応したSolicited-Nodeマルチキャストアドレス（11.3節参照）を指定することになります。

一方、近隣不到達性検知でNeighbor Solicitationメッセージを送信する場合には、対象となるIPv6ノードを宛先とするユニキャストになります。

図6.5にNeighbor Solicitationメッセージのフォーマットを示します。

- Type（8ビット）

  Neighbor Solicitationメッセージであることを示す135が入ります。

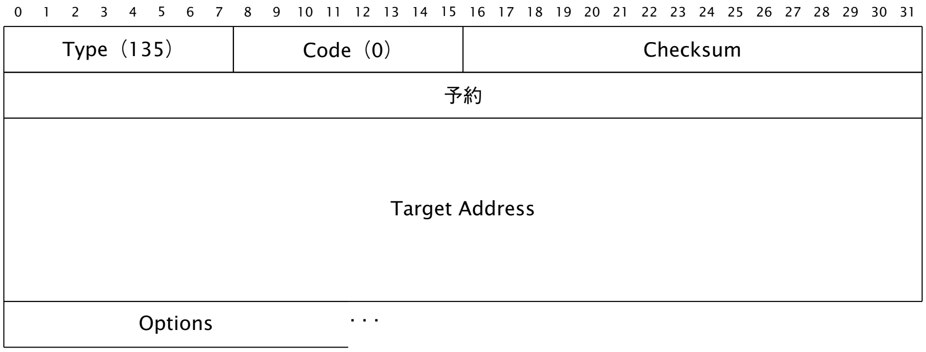
- Code（8ビット）

  0が入ります。

- Checksum（16ビット）

  ICMPチェックサムです。

CC BY NC SA



▶ 図 6.5　Neighbor Solicitation メッセージフォーマット

- 予約（32 ビット）

　未使用の領域です。このフィールドの値は 0 であることが求められます。受信側は、このフィールドの内容を無視する必要があります。

- Target Address（128 ビット）

　対象となる IPv6 アドレスです。このフィールドにマルチキャストアドレスを記載することは禁止されています。

- Options（可変長）

　近隣探索メッセージのためのオプション（6.5 節で説明します）を含めるフィールドです。Neighbor Solicitation メッセージには、Source Link-layer Address オプションを使って、送信者のリンク層アドレスを含めることができます。

6.3.2　Neighbor Advertisement メッセージ

　Neighbor Advertisement メッセージは、Neighbor Solicitation への返答として送信されます。設定情報を更新したときに、それを即座に伝搬させるため、IPv6 ノードが自発的に Neighbor Advertisement メッセージを送信することもあります。

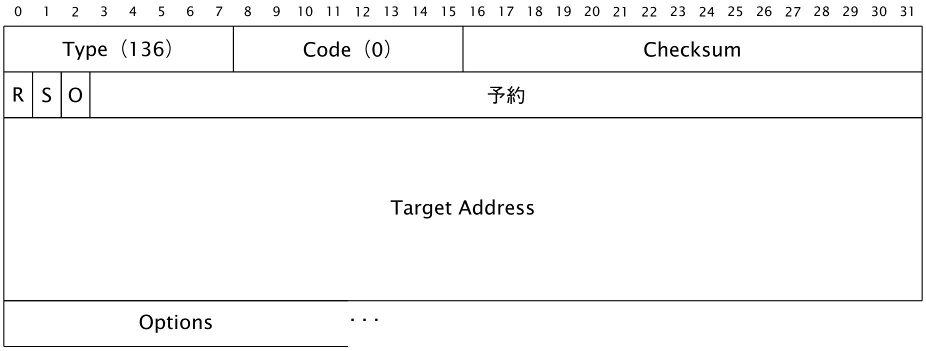
　Neighbor Advertisement メッセージを送信する IPv6 パケットのヘッダに指定する宛先 IPv6 アドレスは、Neighbor Solicitation メッセージへの返答か、それとも自発的な送信かによって異なります。

　Neighbor Solicitation メッセージへの返答の場合は、受け取った Neighbor Solicitation メッセージの送信元アドレスに応じて、返答のための Neighbor Advertisement メッセージの宛先が変わります。Neighbor Solicitation メッセージの送信元 IPv6 アドレスが未定義アドレスではない場合には、Neighbor Solicitation メッセージの送信元アドレスが宛先になります。Neighbor Solicitation メッセージの送信元アドレスが未定義アドレスである場合には、全ノードアドレス（`ff02::1`）を宛先とします。

　自発的に Neighbor Advertisement メッセージを送信する場合は、全ノードアドレス（`ff02::1`）宛に送信します。

Neighbor Advertisementメッセージを送信するIPv6パケットの送信元IPv6アドレスとしては、メッセージを送信するネットワークインターフェースに設定されたIPv6アドレスを指定します。

図6.6にNeighbor Advertisementメッセージのフォーマットを示します。



▶図6.6　Neighbor Advertisementメッセージフォーマット

- Type（8ビット）

  Neighbor Advertisementメッセージであることを示す136が入ります。

- Code（8ビット）

  0が入ります。

- Checksum（16ビット）

  ICMPチェックサムです。

- R（1ビット）

  ルータであることを示すためのフラグです。このフラグが1であるとき、送信元がルータであることを示します。

- S（1ビット）

  Solicitedを意味するフラグで、Neighbor Solicitationメッセージに対する返答であることを示します。近隣不到達性検知では、このフラグを使います。Neighbor Advertisementメッセージの宛先がマルチキャストの場合や、自発的なユニキャスト送信の場合は、Sフラグを1にすることが禁止されています。

- O（1ビット）

  Overrideを意味するフラグで、近隣キャッシュ（6.3.4項参照）に登録されているリンク層アドレス情報の上書きを推奨する場合には1に設定します。このフラグが設定されていない場合、近隣キャッシュに登録済みのエントリがNeighbor Advertisementメッセージの情報で更新されることはありません。エニーキャストでは、このフラグを1にしないことが推奨されています。

- 予約（29ビット）

　未使用の領域です。このフィールドの値は0であることが求められます。受信側は、このフィールドの内容を無視する必要があります。

- Target Address（128ビット）

　Neighbor Solicitationメッセージに対する返答の場合、Neighbor SolicitationメッセージのTarget Addressフィールドに記載されているIPv6アドレスになります。リンク層アドレスの変更を伝える自発的なNeighbor Advertisementメッセージの場合は、このフィールドに、変更のあったリンク層アドレスに対応するIPv6アドレスが入ります。なお、Target AddressフィールドをマルチキャストIPv6アドレスにすることは禁止されています。

- Options（可変長）

　近隣探索メッセージのためのオプション（6.5節で説明します）を含めるフィールドです。

6.3.3　Solicited-Nodeマルチキャストグループへの参加

　近隣探索プロトコルでは、マルチキャストが利用可能なリンクにおいて、アドレス解決や近隣不到達性検知のためにマルチキャストを利用します。マルチキャストが利用可能なネットワークインターフェースが有効になった時点で、ノードは、全ノードアドレスのマルチキャストグループに参加するとともに、Solicited-Nodeマルチキャストグループに参加する必要があります。

　ネットワークインターフェースに設定されたIPv6アドレスが変更される場合には、新しいIPv6アドレスに対応するSolicited-Nodeマルチキャストグループに参加し、古いIPv6アドレスに対応するSolicited-Nodeマルチキャストグループから離脱する必要があります。ただし、1つのSolicited-Nodeマルチキャストアドレスが複数のIPv6アドレスに対応することもあるため、対応するIPv6アドレスが1つでも残っている場合には、そのSolicited-Nodeマルチキャストアドレスから離脱しません。

　Solicited-Nodeマルチキャストアドレスに関しては11.3節で改めて説明します。

6.3.4　Neighbor SolicitationメッセージとNeighbor Advertisementメッセージの送受信

　あるノードが、IPv6アドレスを知っている近隣のノードに対してユニキャストでIPv6パケットを送信しようとしたら、送信先のIPv6アドレスに対応するリンク層アドレスが必要になります。IPv6アドレスからリンク層アドレスへの対応は、**近隣キャッシュ**（Neighbor Cache）として、各ノードで管理しています。

　もしIPv6アドレスに対応するリンク層アドレスが近隣キャッシュになければ、ノードはNeighbor Solicitationメッセージを送信します。その際のIPパケットのヘッダにおける宛先IPv6アドレスは、ネットワークインターフェースがマルチキャストを利用可能なものであれば、Solicited-Nodeマルチキャストアドレス（11.3節参照）にします。

CC BY NC SA

Neighbor Solicitationメッセージによる問い合わせがあると、該当するIPv6アドレスが設定されているネットワークインターフェースを持つノードは、Neighbor Solicitationメッセージを送信したノードに対してユニキャストでNeighbor Advertisementメッセージを返信します。このNeighbor Advertisementメッセージが、Neighbor Solicitationメッセージを送信したノードに届くことで、そのノードはリンク層アドレスを解決できるようになります。

あるノードがNeighbor Solicitationメッセージを送信し、それに対するNeighbor Advertisementメッセージを受け取るまでの流れを、そのノードにおける処理として見てみましょう。ノードでは、リンク層アドレスがわからないIPv6パケットの送信をトリガーとして、Neighbor Solicitationメッセージを送信します。トリガーとなったIPv6パケット自体は、リンク層アドレスが解決されるまでキューに保存します。そして、Neighbor Solicitationメッセージを送信すると同時に、該当するIPv6アドレスのためのエントリを近隣キャッシュに生成します。もちろん、まだIPv6アドレスに対応するリンク層アドレスはわからない状態なので、このエントリは「INCOMPLETE（不完全）」というステートであるとされます（近隣キャッシュのステートについては、6.3.6項で改めて説明します）。

この状態で、ユニキャストのNeighbor Advertisementメッセージを受け取ったら、そのメッセージの情報に対応するエントリが近隣キャッシュにあるかどうか調べます。エントリがあれば、受け取った情報で近隣キャッシュを更新します。もし近隣キャッシュにエントリが存在しなかったら、受け取ったNeighbor Advertisementメッセージは先に送ったNeighbor Solicitationメッセージに対応するものではないとわかります。そのため、受け取ったNeighbor Advertisementメッセージを破棄します。Neighbor Advertisementメッセージを受け取ったからといって、近隣キャッシュに新たなエントリを生成するわけではありません。

ホストごとに設定された時間だけ待ってもNeighbor Solicitationメッセージに対して応答がなかったら、Neighbor Solicitationメッセージを再送します。一定の回数だけ再送してもNeighbor Advertisementメッセージを受け取れない場合は、キューに保存していたIPv6パケットに対してICMP Destination Unreachableメッセージを Code 3（Address Unreachable）で返します。

6.3.5　近隣不到達性検知

ネットワークでは、さまざまな理由で、通信相手までIPv6パケットが届かない状態になってしまうことがあります。宛先ノードそのものが通信不能な状況もあれば、途中経路の障害を回避することで通信できる可能性が残されている場合もあるでしょう。ホストとホスト間、ホストとルータ間、ルータとルータ間など、通信経路上のあらゆる箇所で近隣ノードに到達できるかどうかを調べるための方法が、RFC 4861に定義されている**近隣不到達性検知**（Neighbor Unreachability Detection、**NUD**）です。

NOTE

ルータとルータ間の経路に関しては、ルーティングプロトコルなどによって到達可能性の検知が代替可能なら、ここで説明する近隣不到達性検知を使う必要はありません。

CC BY NC SA

近隣不到達性検知では、自ノードから送ったIPv6パケットを受け取ったことを知らせる何らかの通知を相手ノードから受け取ったときに、その相手ノードへは到達可能であると判断します。この判断に利用できる通知としては、次の2種類があります。

- 上位層のプロトコルにおける応答
- Neighbor Solicitationメッセージに対するNeighbor Advertisementメッセージの応答

1つめの上位層のプロトコルにおける応答としては、たとえばTCPで通信相手のノードからACKを受け取った場合が挙げられます。TCP接続を行っている相手からの応答が受け取れているのであれば、その宛先に到達するために経由する近隣ノードが次ホップとして機能することを示しています。

もう1つが、Neighbor Solicitationメッセージに対する応答としてNeighbor Advertisementメッセージを受け取ったときです。上位層プロトコルで相手への到達可能性が判断できない場合には、自発的に送信したNeighbor Solicitationメッセージへの応答となるNeighbor Advertisementメッセージを受け取ることで、近隣ノードが次ホップとして機能するかどうかを検知します。

Neighbor AdvertisementメッセージのSフラグが1に設定されていれば、こちらから送ったNeighbor Solicitationメッセージが相手に届いたことに対する応答だとわかるので、近隣ノードへの到達性があると確認できます。その一方で、Sフラグが0に設定されているNeighbor Advertisementメッセージを受け取ったり、Router Advertisementメッセージを受け取ったりしても、それらの近隣探索メッセージの送信元ノードへの到達性を確認するものとみなされません。そのようなメッセージは、近隣ノードから自ノードへの到達性を示すものではありますが、自ノードから近隣ノードに対するパケットが到達していることを示しているわけではないからです。近隣不到達性にとって問題になるのは、あくまでも「自ノードから次ホップとなる近隣ノードへ到達できなくなっていないかどうか」なのです。

NOTE

近隣不到達性検知は、ユニキャストでIPv6パケットを送信するときに同時に実施されるものです。宛先がマルチキャストアドレスの場合には実施されません。

6.3.6 近隣キャッシュのステート

近隣キャッシュには、近隣不到達性検知の仕組みを通じて管理される情報も含まれています。それは、近隣への到達可能性を示す以下の5種類の「ステート（state）」です。

- INCOMPLETE

  INCOMPLETEは、不完全という意味を持つ英単語です。該当するエントリに対してアドレス解決が進行中であることを示しています。Solicited-nodeマルチキャストアドレスに対してNeighbor Solicitationメッセージが送信され、Neighbor Advertisementメッセージの応答が受け取れていないときに、このステートになります。

CC BY NC SA

- REACHABLE

  REACHABLEは、到達可能という意味を持つ英単語です。該当するエントリに対応する近隣ノードへの経路が正常であることを示しています。決められた時間内に、近隣ノードへの到達性が肯定的に確認できる通知を受け取れているとき、そのノードに対応するエントリのステートはREACHABLEになります。ステートがREACHABLEのときは、IPv6パケットを送信するときに近隣探索プロトコルの処理は発生しません。

- STALE

  STALEは、新鮮ではない、腐りかけている、古い、などの意味を持つ英単語です。該当するエントリに対応する近隣ノードは到達可能とはいえないことを示しています。近隣ノードへの到達性を肯定的に確認できる通知を受け取ってから所定の時間（ReachableTime）が経過すると、そのノードに対応するエントリのステートはSTALEになります。ステートがSTALEになっても、次にパケットが送信されるまでは、近隣探索プロトコルとしての処理は特に発生しません。

  自発的に送信したNeighbor Solicitationメッセージに対する応答ではない近隣探索メッセージを受け取ったとしも、そのメッセージを送信したノードへの到達性を確認したとはみなされません。そのようなメッセージは、相手ノードから自ノードへの到達性は示しますが、自ノードから送信されたパケットに対する反応でなければ到達性の確認にはならないからです。

- DELAY

  DELAYは、遅延という意味を持つ英単語です。「近隣ノードへの到達性を肯定的に確認できる通知を受け取ってから所定の時間（ReachableTime）が経過してしまったが、送信済みのパケットがあるので、それを待機している期間である」ことを示しています。該当するエントリに対応する近隣ノードは到達可能とはいえないものの、送出済みのパケットがあるので、上位層を通じて肯定的な確認が得られる可能性を考慮して待っている状態です。

  送信済みパケットの通知を待機する期間（通知を待つ時間とは別に決められています）に近隣ノードへの到達性を示す肯定的な確認が得られたら、REACHABLEに移行します。得られない場合は、Neighbor Solicitationメッセージを送信したうえでPROBEへと移行します。

- PROBE

  PROBEは、調べる、探査する、探る、などの意味を持つ英単語です。到達性を確認する応答を待っている状態です。ノードに設定されているNeighbor Solicitationメッセージの再送タイマーごとにNeighbor Solicitationメッセージが再送信されます。

前述したように、IPv6パケットの送信時にリンク層アドレスがわからない場合は、リンク層アドレスの解決を行いつつ、近隣キャッシュにINCOMPLETEステートのエントリを生成します。このエントリは、リンク層アドレスの解決に成功すればステートがREACHABLEに変わります。リンク層アドレスの解決に失敗した場合はエントリごと削除します。INCOMPLETEか

cc BY NC SA

らREACHABLEへの遷移は、Neighbor Solicitationへの応答としてNeighbor Advertisementを受け取った場合のみです。

エントリがSTALE、DELAY、PROBEの各ステートのときに、近隣ノードへの到達性が確認された場合には、ステートがREACHABLEに変わります。STALE、DELAY、PROBEの各ステートからREACHABLEへの遷移では、INCOMPLETEからの場合とは違い、上位層での確認による場合があります。

NOTE

INCOMPLETEは、該当するリンク層アドレスを解決するためのNeighbor Advertisementを一度も受け取っていない状態です。そのため、上位層での確認ではREACHABLEになりません。

近隣不到達性検知におけるReachableTime

近隣不到達性検知では、近隣ノードへの到達性を肯定的に確認できる通知を受け取ってから、ある所定の期間だけ、そのノードが到達可能である（REACHABLE）であるとみなします。この期間は、RFC 4861では「ReachableTime」というミリ秒単位の値で示されています。

ReachableTimeとしては、Router AdvertisementメッセージのReachable Timeフィールドの値をもとに算出することが推奨されています。具体的には、このフィールドの値が0でない場合、その値に対してノード内で生成されたランダムな数値をかけたものをReachableTimeとして利用することが推奨されています（Router Advertisementメッセージが0の場合などに使うデフォルト値も規定されています）。

ReachableTimeは、新しく受け取ったRouter AdvertisementメッセージでReachable Timeフィールドが変化した場合には再計算されます。Router AdvertisementメッセージのReachable Timeフィールドに変化がない場合でも、数時間おきにノード内のReachableTimeの再計算を行うことが推奨されています。

6.4 Redirectメッセージ

近隣探索プロトコルのRedirectメッセージは、他のルータが特定の経路としてより良い次ホップであることを伝えるときや、特定の宛先が実際には同一リンク上に接続された近隣ノードであることを伝えるときに使われます。

RedirectメッセージのIPv6ヘッダに記載される送信元IPv6アドレスは、Redirectメッセージが送信されるネットワークインターフェースのリンクローカルユニキャストアドレスである必要があります。

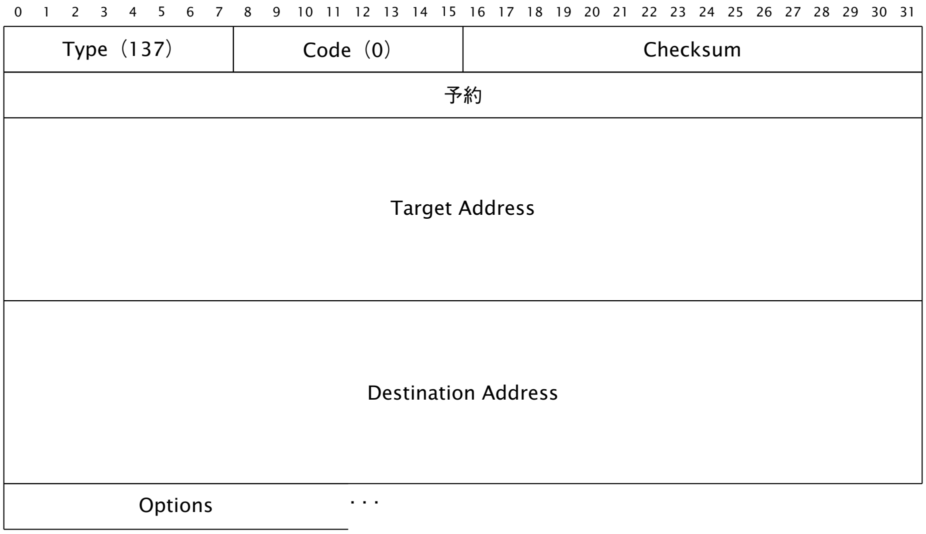
RedirectメッセージのIPv6ヘッダに記載される宛先IPv6アドレスは、ルータがRedirectメッセージを送信するきっかけとなったパケットの送信元IPv6アドレスです。

図6.7にRedirectメッセージのフォーマットを示します。

- Type（8ビット）

  Redirectメッセージであることを示す137が入ります。

CC BY NC SA



▶ 図6.7　Redirectメッセージフォーマット

- Code（8ビット）

　0が入ります。

- Checksum（16ビット）

　ICMPチェックサムです。

- 予約（32ビット）

　未使用の領域です。このフィールドの値は0であることが求められます。受信側は、このフィールドの内容を無視する必要があります。

- Target Address（128ビット）

　より良い次ホップのIPv6アドレスです。Redirectメッセージを送信するきっかけとなった宛先IPv6アドレスが近隣ノードである場合、Target AddressフィールドとDestination Addressフィールドの内容が同じになります。より良い次ホップとしてルータを示す場合には、Target Addressをリダイレクト先となるルータのリンクローカルユニキャストアドレスにする必要があります。

- Destination Address（128ビット）

　リダイレクト先を通じて転送されるべき宛先IPv6アドレスです。

NOTE

Target AddressとDestination Addressの両方のIPv6アドレスを同じものにすることで、特定の宛先が実際には同一リンク上に接続された近隣ノードであることを示せます。

- Options（可変長）

　オプションを含むためのフィールドです。

6.4.1 無効なRedirectメッセージ

以下の要素を満たさないRedirectメッセージは無効なものとして破棄されます。

- ICMPv6メッセージのIPv6ヘッダに記載された送信元IPv6アドレスがリンクローカルアドレスであること
- ICMPv6メッセージのIPv6ヘッダのHop Limitの値が255であること（ルータによって転送されたRedirectメッセージは255よりも小さな値になってしまう）
- ICMPチェックサムが正しいこと
- ICMP Codeが0であること
- そのメッセージを受け取ったノード内の経路表において、宛先と次ホップがICMPv6 Redirectメッセージにある内容と整合性が取れていること
- ICMP Destination Addressフィールドにマルチキャストアドレスが含まれていないこと
- ICMP Target Addressがリンクローカルアドレス、もしくは、ICMP Destination Addressの内容と同じものであること
- ICMPv6メッセージに含まれるすべてのオプションが0よりも大きなサイズであること

6.5 近隣探索メッセージのオプション

RFC 4861で定義されている近隣探索に関するメッセージ（Router Solicitationメッセージ、Router Advertisementメッセージ、Neighbor Solicitationメッセージ、Neighbor Advertisementメッセージ、Redirectメッセージ）には、0個以上のオプションを含めることができます。本書執筆時点で定義されているオプションの一覧を表6.1に示します。

表6.1に示したオプションのうち、本書では、RFC 4861で定義されているSource Link-layer Addressオプション（タイプ1）、Target Link-layer Addressオプション（タイプ2）、Prefix Informationオプション（タイプ3）、Redirected Headerオプション（タイプ4）、MTUオプション（タイプ5）の5つと、RFC 8106で定義されているRDNSSオプション（タイプ25）、DNSSLオプション（タイプ31）の2つについて説明します。

▶ 表6.1 近隣探索メッセージのオプション

Type	オプション名	RFC
1	Source Link-layer Address	RFC 4861
2	Target Link-layer Address	RFC 4861
3	Prefix Information	RFC 4861
4	Redirected Header	RFC 4861
5	MTU	RFC 4861
6	NBMA Shortcut Limit Option	RFC 2491[†11]
7	Advertisement Interval Option	RFC 6275[†12]

[†11] RFC 2491：G. Armitage, P. Schulter, M. Jork, G. Harter, “IPv6 over Non-Broadcast Multiple Access (NBMA) networks”, 1999年1月

[†12] RFC 6275：C. Perkins, D. Johnson, J. Arkko, “Mobility Support in IPv6”, 2011年7月

8	Home Agent Information Option	RFC 6275
9	Source Address List	RFC 3122[†13]
10	Target Address List	RFC 3122
11	CGA option	RFC 3971[†14]
12	RSA Signature option	RFC 3971
13	Timestamp option	RFC 3971
14	Nonce option	RFC 3971
15	Trust Anchor option	RFC 3971
16	Certificate option	RFC 3971
17	IP Address/Prefix Option	RFC 5568[†15]
18	New Router Prefix Information Option	RFC 4068[†16]
19	Link-layer Address Option	RFC 5568
20	Neighbor Advertisement Acknowledgment Option	RFC 5568
21-22	未割当	
23	MAP Option	RFC 4140[†17]
24	Route Information Option	RFC 4191
25	RDNSS	RFC 8106
26	Router Advertisement Flags Extension Option	RFC 5175
27	Handover Key Request Option	RFC 5269[†18]
28	Handover Key Reply Option	RFC 5269
29	Handover Assist Information Option	RFC 5271[†19]
30	Mobile Node Identifier Option	RFC 5271
31	DNSSL	RFC 8106
32	Proxy Signature (PS)	RFC 6496[†20]
33	Address Registration Option	RFC 6775[†21]
34	6LoWPAN Context Option	RFC 6775
35	Authoritative Border Router Option	RFC 6775
36	6LoWPAN Capability Indication Option (6CIO)	RFC 7400[†22]
37	DHCP Captive-Portal	RFC 7710[†23]

†13 RFC 3122：A. Conta, “Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification”, 2001 年 6 月

†14 RFC 3971：J. Arkko, J. Kempf, B. Zill, P. Nikander, “SEcure Neighbor Discovery (SEND)”, 2005 年 3 月

†15 RFC 5568：R. Koodli, “Mobile IPv6 Fast Handovers”, 2009 年 7 月

†16 RFC 4068：R. Koodli, “Fast Handovers for Mobile IPv6”, 2005 年 7 月

†17 RFC 4140：H. Soliman, C. Castelluccia, K. El Malki, L. Bellier, “Hierarchical Mobile IPv6 Mobility Management (HMIPv6)”, 2005 年 8 月

†18 RFC 5269：J. Kempf, R. Koodli, “Distributing a Symmetric Fast Mobile IPv6 (FMIPv6) Handover Key Using SEcure Neighbor Discovery (SEND)”, 2008 年 6 月

†19 RFC 5271：H. Yokota, G. Dommety, “Mobile IPv6 Fast Handovers for 3G CDMA Networks”, 2008 年 6 月

†20 RFC 6496：S. Krishnan, J. Laganier, M. Bonola, A. Garcia-Martinez, “Secure Proxy ND Support for SEcure Neighbor Discovery (SEND)”, 2012 年 2 月

†21 RFC 6775：Z. Shelby, S. Chakrabarti, E. Nordmark, C. Bormann, “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)”, 2012 年 11 月

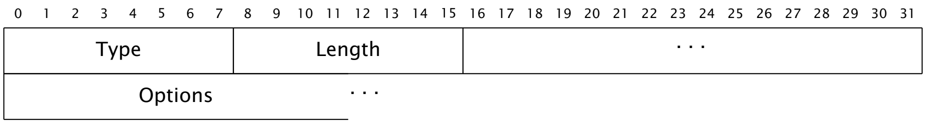
†22 RFC 7400：C. Bormann, “6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)”, 2014 年 11 月

†23 RFC 7710：W. Kumari, O. Gudmundsson, P. Ebersman, S. Sheng, “Captive-Portal Identification Using DHCP or Router Advertisements (RAs)”, 2015 年 12 月

38-137	未割当	
138	CARD Request option	RFC 4065[†24]
139	CARD Reply option	RFC 4065
140-252	未割当	
253	RFC 3692[†25]-style Experiment 1	RFC 4727[†26]
254	RFC 3692-style Experiment 2	RFC 4727

6.5.1 近隣探索メッセージのオプションの基本フォーマット

近隣探索に関するオプションは、いずれも図6.8に示す基本的なTLV構造をしています。



▶図6.8 オプションの基本フォーマット

- Type（8ビット）

  オプションのタイプを指定するフィールドです。指定できる値については表6.1を参照してください。

- Length（8ビット）

  TypeフィールドとLengthフィールドを含めたオプション全体の長さを8ビットの符合なし整数で示します。8オクテット単位で示すので、たとえばLengthの値が2であれば、オプション全体の長さは16オクテットということになります。このフィールドに0を指定することは禁止されており、Lengthフィールドが0になっているオプションを受け取ったノードはパケット全体を破棄する必要があります。

RFC 4861では、オプションが64ビット境界で終端するように、必要に応じてパディングすることが推奨されています。

6.5.2 Source Link-layer AddressオプションとTarget Link-layer Addressオプション

Source Link-layer Addressオプションには、パケット送信者のリンク層アドレスを含めます。Neighbor Solicitationメッセージ、Router Solicitationメッセージ、Router Advertisementメッセージで利用されるオプションです。

---

†24 RFC 4065：J. Kempf, “Instructions for Seamoby and Experimental Mobility Protocol IANA Allocations”, 2005年7月

†25 RFC 3692：T. Narten, “Assigning Experimental and Testing Numbers Considered Useful”, 2004年1月

†26 RFC 4727：B. Fenner, “Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers”, 2006年11月

Target Link-layer Addressオプションには、メッセージの対象となるノードのリンク層アドレスを含めます。Neighbor AdvertisementメッセージとRedirectメッセージで利用されるオプションです。

Source Link-layer AddressオプションとTarget Link-layer Addressオプションは、いずれも同じ図6.9に示すフォーマットをしています。

0–7	8–15	16–31
Type（1または2）	Length	Link-layer Address
Options	…	

▶ 図6.9　Target Link-layer Addressオプション

- Type（8ビット）

  Source Link-layer Addressオプションの場合は1、Target Link-layer Addressオプションの場合は2です。

- Length（8ビット）

  オプション全体の長さを8オクテット単位で示します。リンク層がイーサネットの場合、このフィールドの値は1になります。

- Link-layer Address（可変長）

  リンク層アドレスを格納するためのフィールドです。このフィールドの長さはリンク層の種類によって変わります。

RFC 2464[†27]に、リンク層がイーサネットである場合のTarget Link-layer Addressオプションの例が紹介されています。図6.10のように、48ビット（6オクテット）のIEEE 802アドレスがTypeとLengthフィールドの直後に続きます。

0–7	8–15	16–31
2	1	イーサネットアドレス
イーサネットアドレス（つづき）		

▶ 図6.10　Target Link-layer Addressオプション（リンク層がイーサネットの場合）

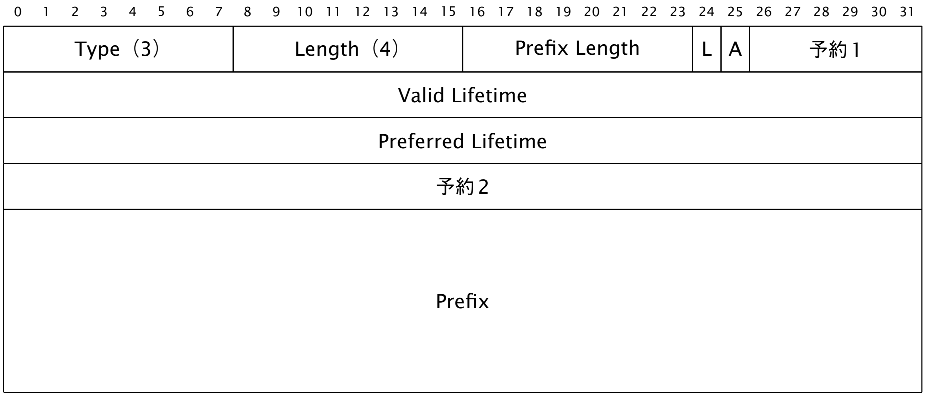
6.5.3　Prefix Informationオプション

Prefix Informationオプションは、on-linkプレフィックスを広告するためのものです。Prefix InformationオプションはRouter Advertisementメッセージのみで利用されます。

- Type（8ビット）

  Prefix Informationオプションを表す3を指定します。

[†27] RFC 2464：M. Crawford, “Transmission of IPv6 Packets over Ethernet Networks”, 1998年12月



▶ 図6.11　Prefix Information オプション

- Length（8ビット）

　オプション全体の長さを8オクテット単位で示します。Prefix Information オプションの長さは、Type フィールドと Length フィールドを含めて32オクテットなので、このフィールドの値は4になります。

- Prefix Length（8ビット）

　プレフィックス長を示す8ビットの符合なし整数です。0から128の値をとります。

- L（1ビット）

　on-link のアドレスであることを示すフラグです（6.6.1項参照）。

- A（1ビット）

　ホストに対し、グローバルアドレスを SLAAC によって設定するように伝えるためのフラグです。ホストは、このフラグが1に設定されているとき、このオプションを含む Router Advertisement メッセージによって示すプレフィックスの値と、ホストが生成するインターフェース識別子を組み合わせて、グローバルアドレスを生成できます。

- 予約1（6ビット）

　未使用領域です。このフィールドの値は0であることが求められます。

- Valid Lifetime（32ビット）

　このプレフィックスがリンク上で有効となる時間を秒単位で示す、32ビットの符合なし整数です。32ビットすべてが1である場合、無期限であることを示します。

- Preferred Lifetime（32ビット）

　SLAAC によって生成されたアドレスが推奨される時間を秒単位で示す、32ビットの符合なし整数です。Valid Lifetime を超えない値であることが求められます。

- 予約2（32ビット）

　32ビットの未使用領域です。このフィールドの値は0であることが求められます。

- Prefix（128ビット）
  IPアドレスもしくはIPアドレスのプレフィックスです。

6.5.4　Redirected Headerオプション

Redirected Headerオプションは、Redirectメッセージを誘発する原因となったパケットの全部もしくは一部を示すために利用されます。このオプションは、Redirectメッセージでのみ利用されます。

0–7	8–15	16–31
Type（4）	Length	予約
予約		
IP header + data ⋮		

▶ 図6.12　Redirected Headerオプション

- Type（8ビット）
  Redirected Headerオプションを表す4を指定します。
- Length（8ビット）
  オプション全体の長さを8オクテット単位で示します。
- 予約（48ビット）
  未使用領域です。このフィールドの値は0であることが求められます。
- IP header + data（可変長）
  Redirectメッセージを誘発する原因となったパケットの全部もしくは一部です。Redirectメッセージの全長がIPv6の最小MTU（1280オクテット）を超えない範囲で、原因となったパケットを切り詰めて格納します。

6.5.5　MTUオプション

同一リンク上のすべてのノードが同じMTUを使うようにするためにRouter Advertisementメッセージに追加できるオプションです。Router Advertisementメッセージ以外でこのオプションが使われることはありません。

0–7	8–15	16–31
Type（5）	Length（1）	予約
MTU		

▶ 図6.13　MTUオプション

- Type（8ビット）

  MTUオプションを表す5を指定します。

- Length（8ビット）

  オプション全体の長さを8オクテット単位で示します。MTUオプションの長さは、Typeフィールドと Lengthフィールドを含めて8オクテットなので、このフィールドの値は1になります。

- 予約（16ビット）

  未使用の領域です。このフィールドの値は0であることが求められます。

- MTU

  リンクで推奨されるMTUです。32ビットの符合なし整数で指定します。

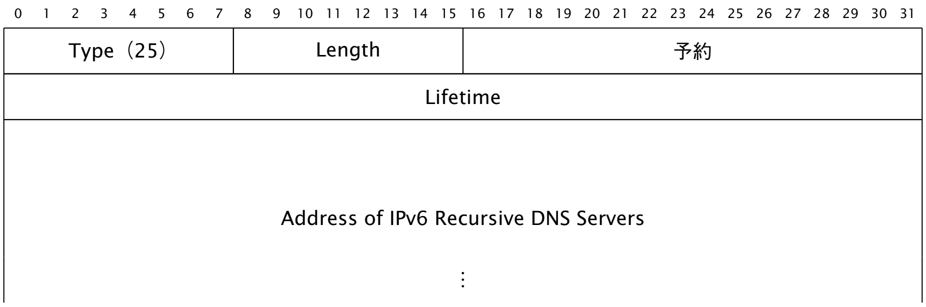
6.5.6　RDNSS（Recursive DNS Server）オプション

IPv6が開発された当初は、Router Advertisementメッセージはあくまでも IPv6アドレスそのものに関する情報を扱うものであり、名前解決のための情報はICMPv6では扱わないものとされていました。IPv6アドレスの設定と名前解決は異なる層の問題であり、IPv6と同じ層の情報を扱うべきICMPv6で名前解決に関する設定を行うのはレイヤーバイオレーションである、というのが主な理由です。当時プロトコル設計にかかわった人々の美意識による決断だったといえるでしょう。

しかし、実際の運用では名前解決も重要であることから、Router Advertisementメッセージに名前解決に関連する情報を扱う機能を入れたいという要望が相次ぎ、ついにはRFC化されました。本書執筆段階では、名前解決に関連するRouter Advertisementメッセージの機能はRFC 8106で定義されています。

RFC 8106で定義されているのは、RDNSSオプションとDNSSLオプションです。これらのオプションは、Router Advertisementメッセージの後半のオプションフィールドに含められます。

RDNSSオプションは、図6.14のように、1つ以上のキャッシュDNSサーバのIPv6アドレスとその有効時間というシンプルな構造になっています。



▶図6.14　RFC 8106 RDNSS

- Type（8ビット）

  RDNSSオプションであることを示す25を指定します。

- Length（8ビット）

  オプション全体の長さを8オクテット単位で指定します。キャッシュDNSサーバのIPv6アドレスが1つであれば、TypeフィールドとLengthフィールドを含めたRDNSSオプションの長さは24オクテットなので、Lengthフィールドの値は3になります。キャッシュDNSサーバのIPv6アドレスが1つ増えるごとに、このフィールドの値は2ずつ大きくなります。

- 予約（16ビット）

  未使用の領域です。このフィールドの値は0であることが求められます。

- Lifetime（32ビット）

  このオプションが示すDNSサーバを使ってもよい期間を秒単位で示す、32ビットの符合なし整数です。すべてのビットが1（`0xffffffff`）であれば、無期限であることを示します。0は記載されたキャッシュDNSサーバをそれ以降使用してはならないことを示します。

- Address of IPv6 Recursive DNS Servers（可変長）

  DNSサーバのIPv6アドレスです。このフィールドに含まれるIPv6アドレスの数は、Lengthフィールドの値から求めます。

1つのRDNSSオプションで、複数のキャッシュDNSサーバIPv6アドレスに対する個別のLifetimeを表現することはできません。しかし、1つのRouter Advertisementメッセージに、それぞれLifetimeが異なる複数のRDNSSオプションを添付することは可能です。

NOTE

RFC 8106のRDNSSオプションの項目には、「RDNSSオプションに含まれるIPv6アドレスがリンクローカルアドレスである可能性を考慮すべき」と記載されています。具体的には、RDNSSオプションの中身がリンクローカルアドレスである場合、それが含まれるRouter Advertisementメッセージを受け取ったリンクに関連するゾーン識別子をリンクローカルアドレスとともに保持することが推奨されています。なお、DHCPv6のDNS Recursive Name Serversオプションが定義されているRFC 3646[†28]には類似する記述はありませんが、同様の可能性が考えられます。

6.5.7 DNSSL（DNS Search List）オプション

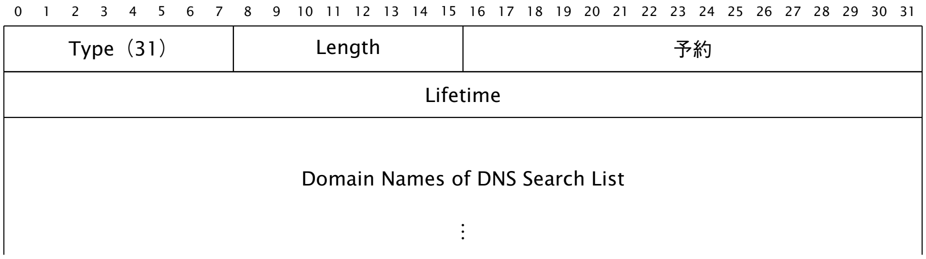
DNSSLは、DNS suffix search listを提供するためのオプションです。たとえば`http://www/`としたときに、DNSSLオプションに`example.com`が設定されていれば、自動的に`http://www.example.com/`であると解釈されたうえで処理されます。「検索ドメイン」「デフォルトのドメイン名」「DNSサフィックス」などの名称で呼ばれることもあります。

---

†28 RFC 3646：R. Droms, “DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2003年12月

実験的なRFCとして発行されたRFC 5006[†29]にはDNSSLオプションは含まれていませんでした。DNSSLオプションが最初に追加されたのはRFC 6106[†30]です。RFC 6106そのものは、2017年3月に発行されたRFC 8106によって上書き廃止されましたが、RFC 8106にもこのオプションが含まれています。

RFC 8106では、DNSSLオプションは図6.15のようなフォーマットになっています。



▶図6.15　RFC 8106 DNSSL

- Type（8ビット）

  DNSSLオプションであることを示す31を指定します。

- Length（8ビット）

  オプション全体の長さを8オクテット単位で示します。このフィールドの値は、Domain Names of DNS Search Listフィールドの長さに応じて変わります。最小値は2です。

- 予約（16ビット）

  未使用の領域です。このフィールドの値は0であることが求められます。

- Lifetime（32ビット）

  このオプションが示すDNSサーバを使ってもよい期間を秒単位で示します。値の意味はRDNSSオプションと同様です。

- Domain Names of DNS Search List（可変長）

  1つ以上のドメイン名を含めます。各ドメイン名は、RFC 1035[†31]のSection 3.1に記載されたフォーマットです。16.3.2項で解説しているフォーマットと同じです。

DNS suffix search listが抱えるセキュリティ上の問題

---

†29 RFC 5006：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Option for DNS Configuration”, 2007年9月

†30 RFC 6106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2010年11月

†31 RFC 1035：P.V. Mockapetris, “Domain names - implementation and specification”, 1987年11月

DNS suffix search listには、まったく異なるFQDN（Fully Qualified Domain Name）へ誘導されるというセキュリティ上の問題があります。この攻撃によって、たとえばパスワードや電子メールのような情報が盗まれる可能性が考えられます。

この問題は、IPv4におけるDHCPにも同様に存在します。IPv4のDHCPにおけるDNS suffix search listのオプションを規定したRFC 3397[†32]では、このような攻撃はDNSSECでも防げないと明記されています。これは、suffixが追加されるのがDNS問い合わせの時点であることから、DNSSECの保護の対象外となるためです。

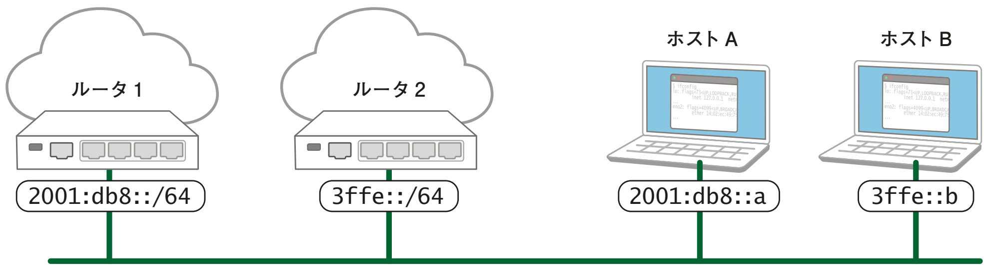
なお、不正なRouter Advertisementメッセージによるセキュリティ上の問題は、DNSSLに限りません。Router Advertisementメッセージに関連するセキュリティについては、14.4節で改めて説明します。

6.6 IPv6におけるon-linkとoff-link

1.2節では、「リンク」および「同一リンク」という概念を説明しました。同一リンクに接続されていることを**on-link**、同一リンクに接続されていないことを**off-link**と表現することがあります。IPv6では、サブネットに対するon-linkとoff-linkのモデルがIPv4とは異なります。

IPv4では、IPv4アドレスとネットマスクを用いて、同一サブネットに属するIPv4アドレスがon-linkであるかどうかを判定できます。しかし、IPv6では、IPv6アドレスとネットワークプレフィックスによりon-linkかどうかを決定することはできません。IPv6で自動的にon-linkであると判定可能なのは、リンクローカルアドレスのみなのです。

IPv6でアドレスとネットワークプレフィックスだけでon-linkを判断できないのは、IPv4と異なり、1つのネットワークインターフェースに複数のIPv6アドレスが設定できるからです。たとえば、図6.16のような、2つのルータと2つのホストからなるネットワークがあるとします。ホストAとホストBの2台の機器が接続されているリンクは、それぞれ別々のネットワークプレフィックス`2001:db8::/64`と`3ffe::/64`をRouter Advertisementメッセージによって広告しているルータ1とルータ2によって同時に運用されています。



▶ 図6.16 2つのルータと2つのホストからなるネットワーク

ホストAのネットワークインターフェースには`2001:db8::a`というIPv6アドレス、ホストBのネットワークインターフェースには`3ffe::b`というIPv6アドレスが設定されていると

†32 RFC 3397：B. Aboba, S. Cheshire, “Dynamic Host Configuration Protocol (DHCP) Domain Search Option”, 2002年11月

CC BY NC SA

しても、ホスト A から 3ffe::b という IPv6 アドレスに対する通信でルータを経由する必要は特にありません。2001:db8::a と 3ffe::b は同一のリンクに接続しているからです。この場合、ホスト A にとって、ホスト B のネットワークインターフェースに設定されている 3ffe::b は on-link であり、on-link のノードに対してはルータを経由させずにパケットを届けることができるはずです。IP アドレスのネットワークプレフィックスが違うというだけでホスト B が off-link であると自動的に判定してしまうと、リンク層ではルータにパケットを転送してもらうような送信方法を採用してしまうでしょう。on-link であるノードに対して直接パケットを送信するには、宛先となる IPv6 アドレスが on-link なのか off-link なのかを判断することが IPv6 では重要な問題になるのです。

NOTE

図 6.16 では、2 つの異なるネットワークプレフィックスを表現するために、例示用の IPv6 アドレス 2001:db::/32 だけではなく 3ffe::/16 を利用しています。3ffe::/16 は、かつて 6bone と呼ばれていた仕組みのために割り当てられていた IPv6 アドレスです。しかし、すでに RFC 3701 によって IANA に返却されており、シスコシステムズ社の各種マニュアルなどでも、3ffe::/16 が例示用のアドレスとして利用されています。そこで、本書でも 3ffe::/16 を例示用として採用することにしました。本書では利用していませんが、5f00::/8 も RFC 3701 によって IANA に返却されており、例示用に使われる場合があります。

6.6.1 Prefix Information オプションの L フラグ

on-link かどうかの判定材料として、Router Advertisement メッセージに含まれる Prefix Information オプション（6.5.3 項参照）があります。ルータは、Router Advertisement メッセージに Prefix Information オプションを含めることで、on-link とみなすプレフィックスをリンク上に広告できます。この Router Advertisement メッセージを受け取ったルータとホストは、ルーティングテーブルに on-link の経路を保持することになります（RFC 4903[†33] の Section 4.2）。

Prefix Information オプションには、そこに含まれる情報を on-link であることの判断に利用してもよいことを示す L フラグが含まれます。L フラグが 1 になっていれば、Prefix Information オプションが示すプレフィックスを on-link の判定に利用してよいということです。L フラグが 0 の場合は、その Router Advertisement メッセージは on-link や off-link に関して何も示さないことを意味します。通常の状態では、L フラグは 1 にセットされます。なお、近隣探索プロトコルを定義している RFC 4861 では、L フラグが 0 であることをもって Prefix Information オプションにより受け取ったプレフィックスが off-link であると判定してはならないとあります。

NOTE

RFC 3756[†34] では、偽の Router Advertisement メッセージによって off-link であるネットワークプレフィックスを on-link であると誤認させるという攻撃の可能性が紹介されています。

---

†33 RFC 4903：D. Thaler, “Multi-Link Subnet Issues”, 2007 年 6 月

CC BY NC SA

6.6.2 Redirectメッセージでon-linkを示すことがある

近隣探索プロトコルのRedirectメッセージは、宛先がon-linkであることを示すために使われることがあります。Redirectメッセージが示すTarget Addressがリンクローカルアドレスではなく、かつDestination AddressがTarget Addressと同じものであるとき、そのメッセージはTarget Addressで示される宛先がon-linkであることを示します。

on-linkであるにもかかわらず、宛先に対してリンク上で直接パケットを送信するのではなく、ルータに対してパケットを送信した場合、ルータはパケット送信元に対してRedirectメッセージを送信します。Redirectメッセージは、宛先がon-linkであるため、ルータではなく直接宛先に対してリンク上でパケットを送信できることを示します。

6.6.3 off-linkであることは明示できない

RFC 5942[†35]は、RFC 4861でも示されているリンクとサブネットの関係について、より明確に解説したものです。IPv6のサブネットモデルとして、on-linkやoff-linkを判定する際のホストの挙動、正しくIPv6が実装されたホストが従うべきルール、間違った実装例などが紹介されています。

RFC 5942によれば、IPv6の近隣探索プロトコルにおいて、ある特定の宛先がoff-linkであることを明確に示すための仕組みは存在しません。on-linkであることが明示された場合のみ、宛先をon-linkとして扱えます。Redirectメッセージでも、off-linkであることを示すようなシグナリングは行いません。Redirectメッセージは、あくまでも次ホップとしてより最適なノードを示すものなのです。

6.6.4 Neighbor Advertisementメッセージはon-linkを示さない

RFC 5942は、IPv6サブネットモデルを解説しているだけではなく、近隣探索プロトコルを定義しているRFC 4861を更新するStandards TrackのRFCでもあります。RFC 4861では次の2点がon-linkの定義とされていましたが、これらはRFC 5942により削除されています。

- そのアドレスからNeighbor Advertisementメッセージを受け取ったこと
- そのアドレスから何らかの近隣探索メッセージを受け取ったこと

これらが削除されたのは、Neighbor Advertisementメッセージに、「本来は遠隔に存在している宛先をon-linkであると誤認させるように偽造することで、パケットの送信先を同一リンク上のノードへと変えられてしまう」という脆弱性がありうるからです（14.2節参照）。

6.6.5 on-link assumptionに関する仕様変更

本書執筆段階における近隣探索プロトコルは、RFC 4861によって定義されています。RFC 4861は、それ以前の近隣探索プロトコルを定義していたRFC 2461を上書き廃止するもの

---

†34 RFC 3756：P. Nikander,J. Kempf,E. Nordmark, “IPv6 Neighbor Discovery (ND) Trust Models and Threats”, 2004年5月

†35 RFC 5942：H. Singh,W. Beebee,E. Nordmark, “IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes”, 2010年7月

CC BY NC SA

です。

RFC 2461 には、「ホストにおけるデフォルトルータのリストが空である場合には、すべての経路を on-link であると想定する」という仕様があります。この仕様は **on-link assumption** と呼ばれます。on-link assumption には、本来であれば経路が存在しないはずのネットワークプレフィックスが on-link であるとの誤認を招き、AAAA レコードと A レコードの両方が DNS に登録された通信相手に対して IPv4 ではなく IPv6 を優先しようとすることで通信開始が遅れるなどの問題があることが RFC 4943 で指摘されています。そのような問題を回避するため、RFC 4861 では、RFC 2461 に含まれていた on-link assumption の仕様が含まれなくなりました。

NOTE

RFC 3756 では、デフォルトルータを「殺す」ことによって意図的にデフォルトルータのリストが空の状態を発生させ、それによりリンク内のホストに対し「すべてのネットワークプレフィックスが on-link である」と誤認させるように誘導する手法が解説されています。

6.6.6 マルチリンクサブネットの問題

マルチリンクサブネットとは、「同一リンクではなく、off-link でありながら、ネットワークプレフィックスは同じになる」という仕組みです。マルチリンクサブネットでは、複数のリンクを 1 つのサブネットとして扱います。IPv6 でマルチリンクサブネットを可能にするための提案が IETF で行われたことがあります。

しかし、本書執筆時点では、IPv6 で複数のリンクにまたがるマルチリンクサブネットは採用されていません。IPv6 アドレス体系を定義している RFC 4291[†36] の Section 2.1 では、IPv6 のサブネットプレフィックスに関して以下のように記述されています。

> Currently, IPv6 continues the IPv4 model in that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link.
> （参考訳）現在、IPv6 では、サブネットプレフィックスは 1 つのリンクであるという IPv4 のモデルを引き続き採用している。複数のサブネットプレフィックスを同一のリンクに対して割り当てることは可能である。

マルチリンクサブネットに関連する問題点は RFC 4903 でまとめられています。そこでは、近隣探索プロトコルのような同一リンク内での利用を想定しているプロトコルが採用している TTL や Hop Limit が想定している挙動にならないおそれがあること、リンクスコープマルチキャストが想定どおりに動作しない可能性があること、リンクスコープマルチキャストの利用を前提としている DAD が正しく動作しない可能性があることなどが紹介されています。また、近隣探索プロトコルに対するセキュリティの仕組みとして考案された SEND（14.3 節参照）を例として、サブネットが単一のリンクによって構成されることを想定しているセキュリティプロトコルが正しく動作しない可能性も紹介されています。

[†36] RFC 4291：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 2006 年 2 月

cc BY NC SA

[書籍版では空白のページです]

第7章

IPv6アドレスの自動設定

IPで通信するには、利用するネットワークインターフェースにIPアドレスを設定する必要があります。IPv4では、当初は手動設定のみを前提としていましたが、あとから作られたDHCP（Dynamic Host Configuration Protocol）を使うことで、IPv4アドレスを動的に自動設定できるようになりました。

これに対し、IPv6では最初からIPv6アドレスを自動的に設定する方法が議論されており、そのための機能がプロトコルそのものに組み込まれています。このIPv6に組み込まれた機能は**SLAAC**（StateLess Address AutoConfiguration、**ステートレスアドレス自動設定**、「すらっく」と発音します）と呼ばれます。

ただし、IPv6でDHCPによる動的なアドレス設定をまったく使わないわけではありません。IPv6で利用できる**DHCPv6**という規格も同時に存在しています。したがって、IPv6アドレスを自動設定する手法としては、SLAACを利用する手法と、DHCPv6を利用する手法の2種類があります。

さらに、一般にインターネットを利用するにはIPアドレスの設定だけではなく、名前解決を行うためのキャッシュDNSサーバのIPアドレスも必要です。DHCPでIPアドレスとキャッシュDNSサーバの両方を同時に設定すればよかったIPv4と違い、IPv6ではキャッシュDNSサーバの自動設定もステートレスな方法とDHCPv6によるステートフルな方法の2種類があります。

ホストによるIPv6アドレスの自動設定方法と、キャッシュDNSサーバの設定方法について整理すると、表7.1の3種類になります。

▶表7.1　IPv6アドレスとDNSサーバの自動設定方法

IPv6アドレスの自動設定	キャッシュDNSサーバの設定
SLAAC	ステートレスDHCPv6
SLAAC	Router Advertisementメッセージ
DHCPv6	ステートフルDHCPv6

CC BY NC SA

NOTE

普通は、表7.1の3種類のうちのどれか1つの組み合わせが使われますが、仕様上は複数の組み合わせが同時に1つのセグメント上で稼働する可能性があります。そのようなとき、たとえばRouter AdvertisementメッセージとDHCPv6とで異なるMTU情報が広告された場合など、設定情報の競合が発生する可能性もあります。また、偽サーバによる設定情報の広告も考えられます。本書執筆時点では、IPv6におけるIPアドレス自動設定に関連する議論は完全に収束しているとはいえず、少しややこしい状態です。

自動設定されるIPv6アドレスは、ネットワークインターフェースに対し、あらかじめ決められた期間だけ貸し出されている（leased）ものです（貸し出し時間が無限の場合もあります）。自動設定されたIPv6アドレスには、そのアドレスが設定されてからの経過時間が記録されており、**存続期間**が切れると他のネットワークインターフェースがそのアドレスを利用する可能性があります。自動設定されるIPv6アドレスが利用可能であるときの状態には、**preferred**（推奨）と**deprecated**（非推奨）の2つがあります。「deprecated」という英単語は、完全に廃止されたという意味で使われることが多い「obsolete」よりも、多少ゆるい表現といえます。実際、自動設定されたIPv6アドレスの存続期間が切れてdeprecatedの状態にあるIPv6アドレスも、使うことが禁止されているわけではありません。SLAACを規定しているRFC 4862[†1]でもそのように明示されており、あくまでもそのまま使うことが「推奨されない」という状態のIPv6アドレスなのです。

NOTE

deprecatedの状態にあるIPv6アドレスを新規のTCP接続などで使う場合には、preferredの状態にあるIPv6アドレスを使うべきです。RFC 4862では、すでに継続しているTCP接続などではdeprecatedの状態にあるアドレスを使い続けてもよいとしています。

本章では、まずSLAACによるIPv6アドレスの自動設定について解説します。それに続き、RFC 8106[†2]（旧RFC 6106[†3]）で追加された、Router Advertisementメッセージによるキャッシュ DNSサーバの設定機能について解説します。DHCPv6については第8章で説明します。

なお、2019年に発行されたIPv6 Node Requirementsを示したRFC 8504[†4]では、RFC 8106によるRDNSSをMUST、ステートレスDHCPv6は「DHCPv6でしか実現できないことがある場合」にはSHOULDとしています。

7.1　SLAACの流れ

SLAAC（StateLess Address AutoConfiguration）は、RFC 4862で定義されているIPv6アドレスの自動生成方法です。RFC 4862では、RFC 4861[†5]で定義されている近隣探索プロトコ

---

†1　RFC 4862：S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration”, 2007年9月

†2　RFC 8106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2017年3月

†3　RFC 6106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2010年11月

†4　RFC 8504：T. Chown, J. Loughney, T. Winters, “IPv6 Node Requirements”, 2019年1月

†5　RFC 4861：T. Narten, E. Nordmark, W. Simpson, H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, 2007年9月

CC BY NC SA

ルなど、他のRFCで定義されている仕組みを組み合わせてSLAACを実現する方法が解説されています。RFC 4862によると、IPv6アドレス自動生成の流れは以下のようになります。

- **リンクローカルアドレスの自動生成**

  MACアドレスなどをもとに各機器でインターフェース識別子を生成し、それにリンクローカルアドレス（3.5節参照）のプレフィックスを付けます。

- **重複アドレスの検知（DAD、Duplicate Address Detection）**

  自動生成したアドレスが有効かどうかを確認します。7.6節で詳しく説明します。

- **グローバルアドレスの自動生成**

  MACアドレスなどをもとに各機器でインターフェース識別子を生成し、それにルータから教えてもらったサブネットのプレフィックスを付けます。

このように、SLAACでは、まずはリンク内での通信を行うためのリンクローカルアドレスを生成し、生成したリンクローカルアドレスが他のノードと衝突していないかを調べたうえで、**グローバルアドレス**（グローバルスコープのIPv6アドレス）を得るための通信を行います。

NOTE

SLAACはIPv6で通信するホストのために設計された仕組みであり、ルータは対象外です。実際、RFC 4862には、「ホストのIPアドレス自動生成がルータからの情報に依存しているため、ルータは別の方法で設定を行う必要がある」と記されています。ただし、リンクローカルアドレスの生成方法と、DADによる重複IPv6アドレスの検知に関しては、ルータについてもRFC 4862の方法を使うとされています。

7.2 小規模ネットワークにおけるSLAACの利用例

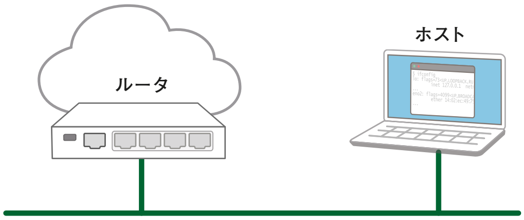
IPv6アドレス自動設定について大まかなイメージを把握するために、小規模なネットワークにおけるSLAACの利用例を紹介します。IPv6で運用されているルータとホストが同一リンクに1台ずつ接続されている図7.1のようなネットワークで、ホストが起動と同時に図7.2のような流れでIPv6アドレスの自動設定を行う状況を説明します。

NOTE

ここで紹介するのは、あくまでも考えられる利用例のひとつです。SLAACに関連する各RFCには、具体的な手順やオプションの組み合わせ例などは記載されておらず、開発者や運用者が柔軟に実装できるようになっています。本書では、あくまでもSLAACの全体像を説明することを目的に、考えられる手順とオプションの組み合わせ例を紹介しています。また、DHCPv6を利用する場合の例については、8.3節で紹介します。

図7.2では、リンクに接続したホストが、まずそのインターフェースに対するリンクローカルアドレスを自動生成します。このときホストが自動生成したリンクローカルアドレスは、この段階ではまだ仮のもの（tentative）という扱いです。Neighbor AdvertisementメッセージとNeighbor Solicitationメッセージを利用したDADにより、同じアドレスを利用している他のインターフェースが同一リンクに存在しないことを確認できたら、初めて正式なリンクロー

CC BY NC SA



▶ 図 7.1　小規模なネットワーク例

カルアドレスとなります。図 7.2 の例では、正式なリンクローカルアドレスが確定後、ホストが自発的な Neighbor Advertisement メッセージをリンクに送信しています。

次にホストは、このインターフェースに対するグローバルアドレスを設定するために、Router Solicitation メッセージを全ルータアドレス（ff02::2）に向けて送信します。これを受け取ったルータでは、このホストに対して Router Advertisement メッセージを返答します。この Router Advertisement メッセージには、このリンクで利用可能なプレフィックス情報がオプションとして含まれています。ホストは、この情報を使って、インターフェースにグローバルアドレスを設定します。こうして自動生成されたグローバルアドレスは、リンクローカルアドレスのときと同様に、DAD 後に正式なものとなります。

NOTE

エニーキャストアドレスに対しては DAD は行われません。また、ホストの設定によってグローバルアドレスに対する DAD が無効になる場合もありますが、RFC 4862 では推奨されていません。

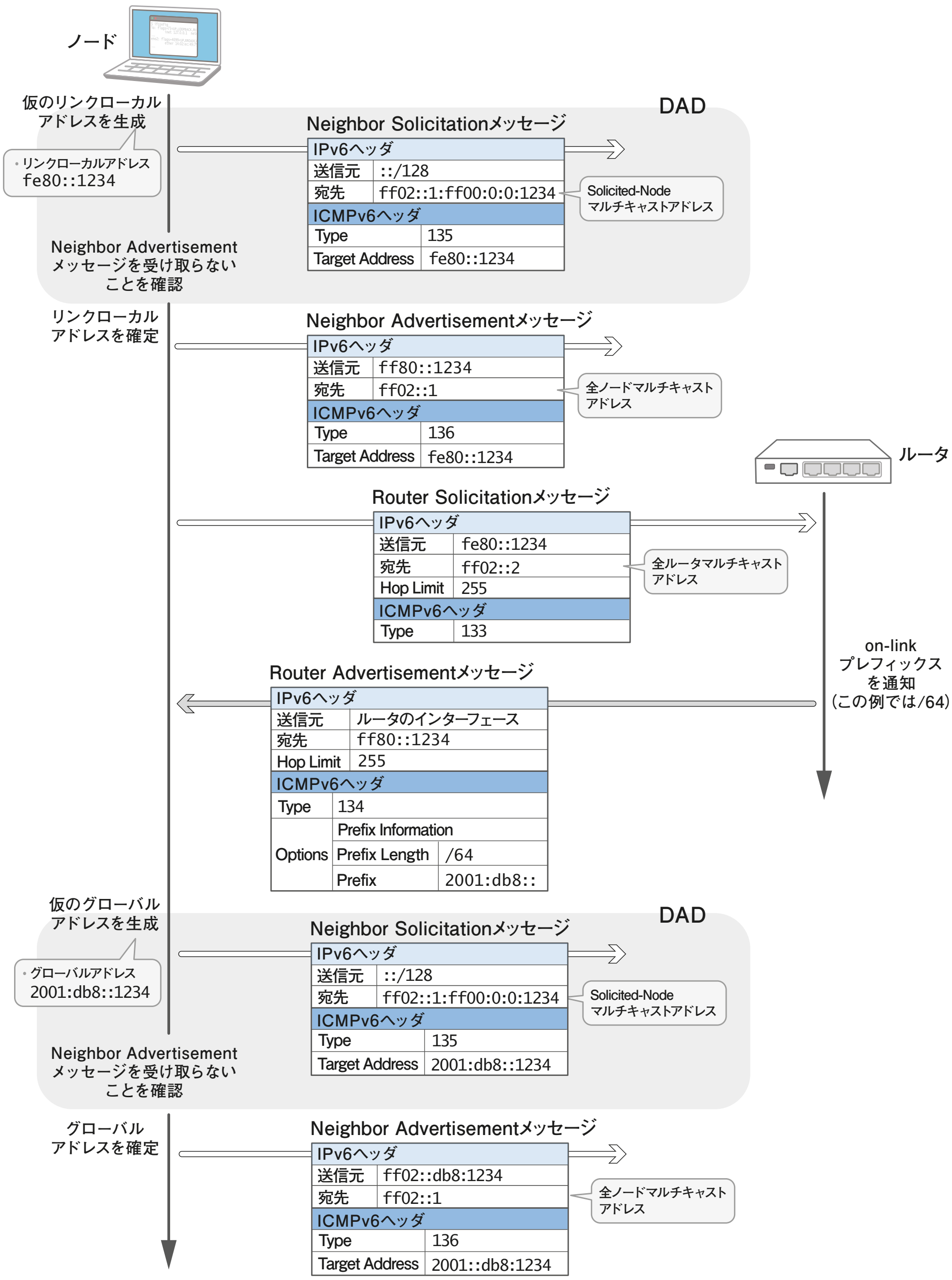
7.3　リンクローカルアドレスの生成

リンクローカルアドレスは同一セグメント内でのみ通信が可能な IPv6 アドレスであり、近隣探索プロトコルなどでも利用される重要な IPv6 アドレスです。リンクローカルアドレスの存続期間は無限であり、タイムアウトによって再生成されるといったことはありません。

IPv6 ノードは、ネットワークインターフェースが有効になると、まずはこのリンクローカルアドレスの生成を試みます。RFC 4862 では、ネットワークインターフェースが有効になるタイミングとして、以下のような状況が想定されています。

- システムが起動する段階でネットワークインターフェースが初期化される場合
- 何らかの理由でネットワークインターフェースに障害が発生し、再度初期化される場合
- システム管理者によってネットワークインターフェースが再度初期化される場合
- ネットワークインターフェースが最初にリンクに接続する場合。たとえば無線ネットワークのアクセスポイントが変化した場合など
- システム管理者によって一時的にネットワークインターフェースが無効化され、再度有効化された場合

(cc) BY NC SA



▶ 図 7.2 小規模なネットワーク例における SLAAC の一例

リンクローカルユニキャストアドレスを定義する RFC 4291[†6]では、リンクローカルユニキャストアドレスのプレフィックスを fe80::/64 としていますが、インターフェース識別子のビット数はリンクの種類と生成方法によって異なるので、SLAAC を定義する RFC 4862 ではリンクローカルアドレスの構成方法について以下のように定めています。

†6 RFC 4291：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 2006 年 2 月

CC BY NC SA

1. IPv6アドレスの左側にある「プレフィックス長」分のビットを、リンクローカルプレフィックスとする
2. リンクローカルプレフィックスの右側部分のビットをすべて0にする
3. インターフェース識別子がnビットなら、IPv6アドレスの右側nビットをインターフェース識別子とする（ただしプレフィックス長とnの合計がIPv6アドレスのビット数である128よりも大きくなってはならない）

上記の3で利用するインターフェース識別子については、いくつか生成方法が考えられます。

7.4　SLAACにおけるインターフェース識別子の生成方法

RFC 4291のSection 2.5.1では、プレフィックス長64ビットで下位64ビットをインターフェース識別子とするとき、下位64ビットの一意性を確保する手段として、**Modified EUI-64形式**を使うことが要求されています。そのうえで、RFC 4291のAppendix Aでは、イーサネットにおけるMACアドレスを使ってModified EUI-64形式のインターフェース識別子を生成する方法が解説されています。

しかし、MACアドレスを利用するModified EUI-64形式には、プライバシーに対する懸念もあります。そのため、乱数に基づくインターフェース識別子からプライバシーを考慮した**Temporary IPv6アドレス**を生成する方法もRFC 8981[†7]で規定されています。また、2014年に発行されたRFC 7217[†8]でも、SLAACでModified EUI-64形式を使わない方法が示されています。

2017年に発行されたRFC 8064[†9]では、SLAAC以外の仕様においても、固定されたインターフェース識別子の扱いがSHOULD NOT（非推奨）とされました。ただし、RFC 8064はModified EUI-64形式そのものを非推奨としているわけではなく、固定されたインターフェース識別子を生成する方法としてRFC 2464で定義されている方法を利用しないことを推奨しているため、たとえばMACアドレス自体が定期的にランダムに変化するような固定ではなくなる場合においてModified EUI-64形式を利用することが非推奨とされているわけではありません。また、MUST NOT（禁止）ではないのでModified EUI-64形式による固定されたインターフェース識別子が完全に消えるわけではなく、実際の運用でも広く利用されていることから、本書でもイーサネットのMACアドレスからModified EUI-64形式のインターフェース識別子を生成する手法を説明しておきます。

NOTE

書籍におけるSLAACの説明でインターフェース識別子の値をどう示すかは、少し悩ましいところがあります。本書では、SLAACを説明する図において、インターフェース識別子を「1234」のように記述することにしました（図7.2など）。本来であれば書籍中の例においても「乱数に

†7　RFC 8981：F. Gont, S. Krishnan, T. Narten, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6”, 2021年2月

†8　RFC 7217：F. Gont, “A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)”, 2014年4月

†9　RFC 8064：F. Gont, A. Cooper, D. Thaler, W. Liu, “Recommendation on Stable IPv6 Interface Identifiers”, 2017年2月

CC BY NC SA

基づくインターフェース識別子」を利用すべきかもしれませんが、図として見にくくなってしまうことや、その値に何らかの意味があるように誤解されてしまう可能性を考え、シンプルな値を例示に使うことにしたものです。

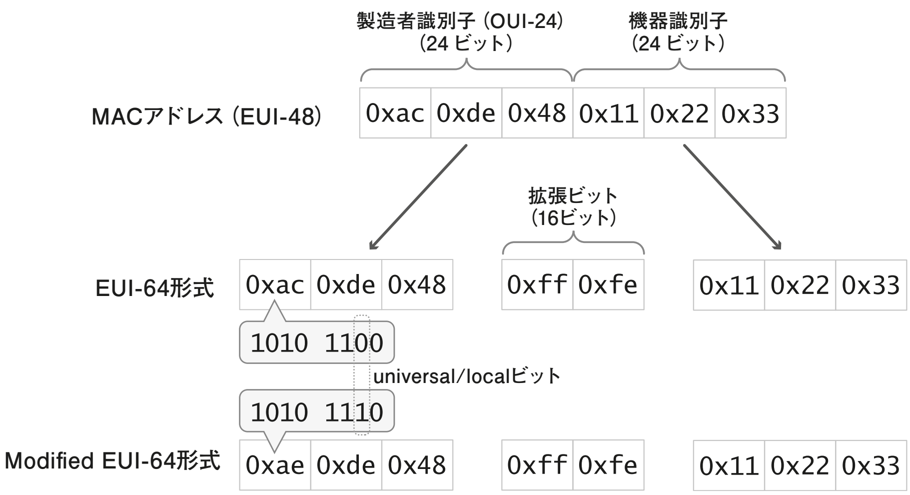
7.4.1 Modified EUI-64形式によるIPv6アドレス生成

IEEE 802で規定されている48ビット長のMACアドレス（EUI-48形式）から、64ビット長のEUI-64形式のアドレスを生成する手法は、IEEEで規定されています。IPv6で利用するModified EUI-64形式と、IEEEで規定されているEUI-64形式との違いは、元のEUI-48に含まれるuビットを反転させる点です。

EUI-48形式では先頭オクテットのうち0x02のビットをuビットと呼び、このビットが0の場合はIEEEによって管理されたユニバーサルなアドレスであることを意味します。逆に、そのビットが1である場合はローカルな管理者によって独自に付けられたアドレスであることを意味します。また、EUI-48形式では先頭オクテットのうち0x01のビットをgビットと呼び、このビットが1であればそのMACアドレスはリンク層におけるマルチキャストアドレスになります。

EUI-48形式のユニバーサルなMACアドレスでは、上位24ビット（3オクテット）がOUI（Organizationally Unique Identifier）と呼ばれる製造者識別子を表します。OUIは、製造者がIEEEに1650米ドルを支払って登録するものであり、一意性が確保されています。EUI-48アドレスの下位24ビットは、各製造者が独自に割り当てられる機器識別子です。

EUI-48形式のアドレスから、IPv6アドレスで利用するModified EUI-64形式のインターフェース識別子を生成するには、図7.3のようにOUI部分のuniversal/localビットを反転させたうえで、OUI部分と機器識別子部分の間に0xfffeという16ビットを挿入します。



▶ 図7.3 EUI-48形式からModified EUI-64形式の生成

OUIのuniversal/localビットは、0がセットされた場合にグローバルに一意な値であることを示し、1がセットされた場合にはローカルな値であることを示しています。このuniversal/localビットを反転させる理由として、RFC 4291のSection 2.5.1では、「ハードウェアが存在しないネットワークインターフェースに対して、ネットワーク運用者がインター

CC BY NC SA

フェース識別子を付けやすいように」という説明が書かれています。OUIでは、グローバルに一意な値である場合にはuniversal/localビットが0なので、`0:0:0:1`のような値はグローバルに一意です。IPv6でグローバルに一意ではないものを示すために`0200:0:0:1`というインターフェース識別子を使うより、IPv6ではuniversal/localビットの示す値の内容を反転させることで、たとえば`0:0:0:1`というインターフェース識別子がグローバルに一意ではないものを示すようにしたわけです。RFC 4291では、Modified EUI-64形式のuniversal/localビットで示される一意性に関して、IPv6ノードが検証することを要求しておらず、その存在理由についても「将来それを利用するテクノロジで活用できるようにするため」とされています。しかし、Modified EUI-64形式は、universal/localビットが実際に使われることがないままRFC 8064で非推奨となりました。

universal/localビットを反転させるのはIPv6用の拡張ですが、`0xfffe`の挿入はIEEEによって規定されたEUI-64の生成方法です。この`0xfffe`は、EUI-48形式をカプセル化してEUI-64形式のアドレスを生成するために予約された値であり、他のEUI-64形式のアドレスと被らないようになっています。

■ RFC 7136での変更

RFC 4291では以下のように規定されているので、通常のユニキャストIPv6アドレスはすべてModified EUI-64形式になっている必要がありました。

> IPv6アドレスの先頭3ビットが000でないすべてのユニキャストIPv6アドレスのインターフェース識別子は64ビット長であり、その部分はModified EUI-64形式によって構成される必要がある。

しかし、2014年に発行されたRFC 7136[†10]では、この部分が次のように修正されています。

> IPv6アドレスの先頭3ビットが000ではないすべてのユニキャストIPv6アドレスのインターフェース識別子は64ビット長である必要がある。もしインターフェース識別子がIEEE MAC層のアドレスから生成されているのであれば、その部分はModified EUI-64形式によって構成されなければならない。

この変更により、プライバシーを考慮したインターフェース識別子や、ISATAP用のインターフェース識別子、4rd用のインターフェース識別子などのIEEE MAC層アドレスとは関係がないインターフェース識別子で、uビットとgビットを扱う必要がなくなりました。

7.5 Temporary IPv6アドレス

Temporary IPv6アドレスは、一定の時間で無効となるランダムな値をインターフェース識別子として自動生成されるIPv6アドレスです。時間と共に新たに作り直すことで、ユーザは同じIPv6アドレスを使い続けることを避けられるため、ネットワークの盗聴やその他方法によってIPv6アドレスをベースとした情報収集による攻撃を軽減することができます。また、

†10 RFC 7136：B. Carpenter, S. Jiang, “Significance of IPv6 Interface Identifiers”, 2014年2月

CC BY NC SA

過去の通信履歴によってIPv6アドレスが外部に暴露されてしまう時間を制限できるという利点もあります。Temporary IPv6 アドレスはRFC 8981で定義されています。

Temporary IPv6 アドレスは、内部から外部への通信を行うための用途を想定しています。外部にIPv6アドレスを公開する必要がある電子メールやWebなどのサーバで使うのではなく、ユーザ側の環境で利用するためのものです。

Temporary IPv6 アドレスは、数時間や数日といった短い時間でdeprecated（非推奨）になります。deprecatedとなる前から続いている通信はTemporary IPv6 アドレスが無効となるまでは利用可能ですが、新たな通信には別のIPv6アドレスが利用されます。deprecatedや期限切れ（無効）となるTemporary IPv6 アドレスを置き換えるための新しいTemporary IPv6 アドレスが生成されます。

Temporary IPv6 アドレスが存続可能な時間は、そのアドレスが生成されるたびに計算されます。これにより、部外者が新しいTemporary IPv6 アドレスの生成時期を予測したり、既存のTemporary IPv6 アドレスと新たに作成されるTemporary IPv6 アドレスを関連づけたりしにくくなります。計算にあたっては、7.8節で説明するpreferred-lifetimeやvalid-lifetimeといった値が用いられます。具体的には、valid-lifetimeとしては「プレフィックスのValid Lifetime」か「TEMP_VALID_LIFETIME（下記参照）」の小さいほうを採用し、preferred-lifetimeとしては「プレフィックスのPreferred Lifetime」か「TEMP_PREFERRED_LIFETIME（下記参照）とDESYNC_FACTORの差」の小さいほうを採用します（DESYNC_FACTORは、生成されるTemporary IPv6 アドレスのpreferred-lifetimeの存続時間がすべて同じにならないようにするためのランダムな値です）。

これらの計算に使うパラメータとそのデフォルト値は、RFC 8981では次のように定められています。なお、RFC 8981により廃止されたRFC 4941[†11]ではこれらのパラメータの扱いが大きく違うので注意してください。

- TEMP_VALID_LIFETIME

  デフォルト値は2日。RFC 8981では、ユーザがこの値を設定可能とすべきとあります。

- TEMP_PREFERRED_LIFETIME

  デフォルト値は1日。RFC 8981では、ユーザがこの値を設定可能とすべきとあります。TEMP_VALID_LIFETIMEよりも小さな値である必要があります。

- REGEN_ADVANCE

  「DADにTEMP_IDGEN_RETRIES回失敗する」という最悪の状況でDADにかかる時間で、以下により計算されます。

  2 ＋ (TEMP_IDGEN_RETRIES × DupAddrDetectTransmits × RetransTimer/1000)

  RetransTimerとDupAddrDetectTransmitsはDADのパラメータで、それぞれRFC 4861、RFC 4862で定義されています（7.6節参照）。REGEN_ADVANCEの単位は秒ですが、RetransTimerの単位はミリ秒なので、この式では最後に1000で割っています。

†11 RFC 4941：T. Narten,R. Draves,S. Krishnan, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”, 2007年9月

cc BY NC SA

このパラメータにより、既存のTemporary IPv6アドレスのpreferred状態が終了するよりも前に、余裕をもって新たなTemporary IPv6アドレスの生成開始を行うことができます。

- MAX_DESYNC_FACTOR

  DESYNC_FACTORの上限で、以下により計算されます。

  0.4 × TEMP_PREFERRED_LIFETIME

  MAX_DESYNC_FACTORをTEMP_PREFERRED_LIFETIMEの0.4倍にすることで、異なる存続時間を持つTemporary IPv6アドレスを生成しつつ、同時に存在するTemporary IPv6アドレスの最大値を3個に制限することができます。

- DESYNC_FACTOR

  0からMAX_DESYNC_FACTORまでのランダムな値。Temporary IPv6アドレスが生成されるたびに計算され、そのアドレスと関連づけされます。TEMP_PREFERRED_LIFETIMEからREGEN_ADVANCEを引いた値よりも小さい値である必要があります。

- TEMP_IDGEN_RETRIES

  REGEN_ADVANCEの計算に使うパラメータです。デフォルト値は3。

RFC 8981では、「既存のTemporary IPv6アドレスがdeprecatedとなるときに新しいTemporary IPv6アドレスを生成する」としています。その一方で、各プレフィックスごとに最低1つはpreferred状態のTemporary IPv6アドレスが存在していることも推奨しています。新たに生成されるTemporary IPv6アドレスに対してはDADが必須なので、既存のTemporary IPv6アドレスがdeprecatedになってから新しいものを生成するのでは、そのDADにかかる時間などで「preferred状態のTemporary IPv6アドレスが1つも存在しない瞬間」が発生する可能性もあります。そうならないように、既存のTemporary IPv6アドレスがdeprecatedとなる少し前に新たなTemporary IPv6アドレスの生成を開始することが推奨されています。

7.5.1　RFC 8981による変更

もともとのTemporary IPv6アドレスは、Stable address[†12]に加えて設定することが想定されていたアドレスです。Stable addressを設定せずにTemporary IPv6アドレスのみを設定することは想定外でした。2021年にRFC 4941を廃止するRFC 8981が発行され、それまで想定外だったTemporary IPv6アドレスのみの利用が可能となっています。

以下にRFC 4941とRFC 8981との差分をまとめます。

- RFC 4941では、Temporary IPv6アドレスはデフォルトで利用しない設定とすることを推奨していましたが、RFC 8981ではその文言が削除されています。
- RFC 4941では、TEMP_VALID_LIFETIMEのデフォルト値を7日間（1週間）としていましたが、RFC 8981では2日に短縮されています。

†12　RFC 7721では、Temporary addressを「同一IPv6リンク内において時間と共に変化するアドレス」、Stable addressを「同一IPv6リンク内において時間と共に変化しないアドレス」と定義しています。本書でも、Temporary addressとStable addressについてはRFC 7721と同様の意味で利用します。

CC BY NC SA

- RFC 4941 およびRFC 8981 ともに、TEMP_PREFERRED_LIFETIMEのデフォルト値は1日としています。
- RFC 4941 では、REGEN_ADVANCEが5秒、MAX_DESYNC_FACTORが10分という定数でしたが、RFC 8981 では他の要素を利用した変数に変わりました。
- RFC 4941 ではシステム起動時に一度だけDESYNC_FACTORを決定するという仕組みでしたが、RFC 8981 ではTemporary IPv6 アドレスを生成するたびに計算し直す仕様に変更されました。
- RFC 4941 では、preferredの状態が約1日なので、deprecated（利用は推奨されないものの無効ではない）な状態のTemporary IPv6 アドレスが6日間存在し続けることになります。この状態になってしまった大量のTemporary IPv6 アドレスが発生することもありました。これに対し、MAX_DESYNC_FACTORなどが工夫されたRFC 8981 の実装では、同じプレフィックスのTemporary IPv6 アドレスが3個以上になりません。deprecatedのまま存在し続けるTemporary IPv6 アドレスの数が減るので、ネットワークで扱うマルチキャストアドレスの数も減らすことができます（Solicited-Nodeマルチキャストアドレスに関しては11.3節参照）。
- RFC 4941 では、Temporary IPv6 アドレスにおける疑似乱数値の生成方法として、履歴情報を保存可能な記憶領域が存在している場合とそうでない場合の2通りの手法が解説されていました。そして、stable storageやhistoryといった値を計算するのに、ハッシュ関数としてMD5 を利用しています。RFC 8981 では、これらの値については言及がなく、MD5 も容認できないとしています。
- RFC 8981 では、セキュリティやプライバシーに関する詳細についてRFC 7721[†13]、RFC 7707[†14]、RFC 7217 を参照する形になっています。RFC 7721、RFC 7707、RFC 7217 に関しては第14章でも説明します。

7.6 DAD（Duplicate Address Detection）

DAD（Duplicate Address Detection）は、IPv6 アドレスをネットワークインターフェースに設定する前に、リンク内で重複するアドレスが利用されていないかを調べるための手順です。SLAAC を定義しているRFC 4862 でDAD も定義されています。

ネットワークインターフェースに対して新たにユニキャストIPv6 アドレスを追加する前には、そのIPv6 アドレスが同一リンク内で他のノードによって使われていないか確かめるため、必ずDAD を実行する必要があります。IPv6 アドレスの設定が行われる方法が、手動設定、SLAAC、DHCPv6 のいずれの方法であってもDAD が必須となることは変わりません。

ただし例外として、以下の状況ではDAD を実行しなくてよい場合もあります。

- DupAddrDetectTransmitsという変数が0に設定されているネットワークインター

†13 RFC 7721：A. Cooper,F. Gont,D. Thaler, “Security and Privacy Considerations for IPv6 Address Generation Mechanisms”, 2016年3月

†14 RFC 7707：F. Gont,T. Chown, “Network Reconnaissance in IPv6 Networks”, 2016年3月

CC BY NC SA

フェースに対しては、DADは実行されない

- エニーキャストでは同じIPv6アドレスで示される宛先がリンク内に同時に複数存在する可能性があるので、エニーキャストアドレスに対してDADを行うことは禁止されている（エニーキャストに関しては第12章参照）
- 実装によってはDADを意図的に実行しないこともある（RFC 4862では推奨されていない）

DADによって、同一リンク内の他ノードが同じIPv6アドレスを使っていないと判断されたら、ネットワークインターフェースに対してIPv6アドレスが設定されます。DADの実行と並行してネットワークインターフェースにIPv6アドレスを有効にすることは禁止されています。並行して新しいIPv6アドレスを設定してしまうと、衝突が発生した際、同じIPv6アドレスで先に運用されていた他ノードのTCPコネクションをリセットしてしまうといった影響があるからです。

DADが失敗した場合、そのIPv6アドレスをネットワークインターフェースで有効にすることは禁止されています。RFC 4862には、もし当該のIPv6アドレスがイーサネットのMACアドレスなどのハードウェアアドレスから生成したEUI-64形式の値であり、リンク上で一意の値であるはずという場合には、そのネットワークインターフェースでIPv6を利用した通信を控えるべきとあります[†15]。IPv6アドレスの重複がハードウェアアドレスの重複を示している場合、他の重複していないIPv6アドレスを設定したとしても下位層での通信に失敗する可能性が高いためです。

なお、DADは完全ではないため、同一リンク内に同じIPv6アドレスが複数登場してしまう可能性もある点に注意が必要です。また、自分自身が送信したNeighbor Solicitationメッセージを自分で受け取ってしまう可能性もあります。

7.6.1 DADにおけるNeighbor SolicitationメッセージとNeighbor Advertisementメッセージの送受信

DADでは、ネットワークインターフェースに設定しようとしているIPv6アドレスが他のノードによって利用されていないことを確認するために、これから設定しようとしているIPv6アドレスをTarget Addressフィールドに設定したNeighbor Solicitationメッセージを送信し、その応答が他ノードからこないことを確認します。Neighbor Solicitationメッセージに対する応答としてNeighbor Advertisementメッセージが届いた場合には、すでに他のノードがそのIPv6アドレスを利用していることがわかります。

DADで送信されるNeighbor Solicitationメッセージでは、Target Addressフィールドには確認したいIPv6アドレスを、IPv6ヘッダに記載される送信元IPv6アドレスには未定義アドレス（`::/128`）を、IPv6ヘッダに記載される宛先IPv6アドレスには確認したいIPv6アドレスに対応するSolicited-Nodeマルチキャストアドレス（11.3節参照）を含めます。

ノードは、Neighbor Solicitationメッセージを送信する前に、全ノードアドレスと、確認

---

†15 ただし、7.4節で述べたように、ハードウェアアドレスを利用したIPv6アドレスの生成は推奨されていません。

CC BY NC SA

したい tentative な IPv6 アドレスに対応する Solicited-Node マルチキャストグループに参加しておく必要があります。全ノードアドレスに参加することで、すでにその IPv6 アドレスを利用している Neighbor Advertisement メッセージを受け取れるようになります。tentative な IPv6 アドレスに対応する Solicited-Node マルチキャストグループに参加することによって、他のノードが同じタイミングで同じ IPv6 アドレスに対する DAD を行っている状態を検知できます。

NOTE

実は、この DAD の段階では、マルチキャストにかかわる ICMPv6 メッセージ（MLD メッセージ）を送信する際の送信元 IPv6 アドレスとして利用可能な IPv6 アドレスが存在しないという問題があります。DAD と MLD に関する課題については 11.5.5 項で扱います。

DAD では、Neighbor Solicitation メッセージを一定の回数だけ送信したあと、あらかじめ設定したミリ秒が経過したことをもって、競合する IPv6 アドレスが同一リンク上に存在しないと判断します。

送信回数を管理する変数は、`DupAddrDetectTransmits` と呼ばれます。この変数はノードで設定が可能です。`DupAddrDetectTransmits` に 0 を設定すれば、DAD を実行しないことを示します。RFC 4862 では、`DupAddrDetectTransmits` のデフォルト値を 1 としています。

待ち時間は、`RetransTimer` という変数で管理します。近隣探索プロトコルを定義している RFC 4861 では、`RetransTimer` 変数の値として、Router Advertisement メッセージに含まれる Retrans Timer フィールドの値をコピーして使うことを推奨しています。RFC 4861 に記載されている Retrans Timer フィールドで使われるデフォルト値は 1000 ミリ秒です。

7.6.2 Enhanced DAD

IPv4 マルチキャストの仕様を示した RFC 1112 では、「自分自身がマルチキャストグループに対して送信しているパケットを自分自身に対して送信する方法」を提供することを推奨しています。これは、同じマルチキャストグループに参加している他のアプリケーションが同一ホスト内に存在する可能性があるため、同一ホスト内でのループバックが存在しなければ、同一ホストから送信されるマルチキャストパケットを受け取れなくなってしまうからです。

IPv6 でも、IPv4 と同様に、マルチキャストで送信されるパケットがループバックされることがあります。DAD の実行時には Neighbor Solicitation メッセージをマルチキャストで送信し、そのマルチキャストグループに参加しているので、自分自身が送信した DAD のための Neighbor Solicitation がループバックで戻ってきてしまい、それを他ノードからの Neighbor Solicitation と誤検知してしまうという問題があります。この問題については RFC 4862 の Appendix A で解説されています。

この問題に対する緩和策として、**Enhanced DAD** という方式が RFC 7527[†16] で定義されています。Enhanced は、改善された、という意味を持つ英単語です。

[†16] RFC 7527：R. Asati,H. Singh,W. Beebee,C. Pignataro,E. Dart,W. George, “Enhanced Duplicate Address Detection”, 2015 年 4 月

CC BY NC SA

RFC 7527では、DADでループバックを検知する必要性があるとしています。それを示すために、ループバックによるDADの誤検知が発生する環境として、RFC 4862よりも具体的な状況が2つ紹介されています。

1つめの事例は、トラブルシューティング用にループバックする回路がネットワーク上に用意されていて、ルータが送出したNeighbor Solicitationメッセージがループバックされることで DADに失敗してしまい、ルータのネットワークインターフェースにIPv6アドレスが割り当てられなくなってしまうという状況です。IPv4ではループバックしている回路を停止すれば通常の動作に戻りますが、IPv6ではDADが失敗した状態から復帰するために手動での介入が必要になってしまうという問題があります。

2つめの事例は、2つのブロードバンドモデムが同じサービスプロバイダに接続しつつ、物理的に1台のルータに収容されており、さらに2つのモデムのイーサネットインターフェースが同じコリジョンドメインとなるローカルネットワークに接続されているという状況です。ルータから送信されたNeighbor Solicitationメッセージが片方のモデムに届くと、そのモデムからもう片方のモデムに対してローカルネットワーク経由でメッセージが転送され、それがそのままサービスプロバイダのルータへと送信されてしまいます。こうして送信されてきたNeighbor Solicitationメッセージにより、ルータがDADに失敗した状態になってしまいますが、この例では当該のルータが数千台のモデムを収容することもあり、それらすべてでIPv6ネットワークの障害を発生させてしまいます。さらにRFC 7527では、このような環境ではどこでループバックが発生しているかをルータ側で把握するのが困難な場合もあると書かれています。

RFC 7527では、Neighbor Solicitationメッセージのループバックに対する緩和策として、DADを実行しないという方法が紹介されています。また、リンク層プロトコルがループバックを検出した際にDADを無効化するという手法も紹介されています。

さらに、Neighbor SolicitationメッセージにランダムなNonceを追加することによって、自分自身が送信したNeighbor Solicitationメッセージかどうかを判別する方法も定義されています。もし同時に複数のノードがまったく同じTarget Addressを生成し、さらにNonceの値も同じNeighbor Solicitationメッセージが送信されれば、本来検出すべきであったIPv6アドレスの競合を見逃してしまうことになりますが、RFC 7527ではランダムな数字の生成の品質が高ければその可能性は低いとしています。Nonceを指定するためのNonceオプションについては14.3節を参照してください。

RFC 7527で定義されているNonceを利用する手法は、近隣探索による任意のSolicitationやAdvertisementにおけるループバックを検出可能です。しかし、すべての近隣探索メッセージに対するNonceや関連するステートを保持することによって実装が複雑になることや、消費メモリを上昇させる可能性もあるため、DAD以外での利用は推奨されないとあります。

7.7　グローバルアドレスの生成

グローバルアドレスは、ICMPv6を利用した近隣探索プロトコルによるRouter Advertisementメッセージによって得られたプレフィックスに、インターフェース識別子を追加するこ

CC BY NC SA

とで生成されます。Router Advertisementメッセージはルータから定期的に送信されますが、グローバルアドレスを生成したいノードがRouter Advertisementメッセージの送信を要求するRouter Solicitationメッセージを送信して即座に受け取ることも可能です。

Router Advertisementメッセージを利用せず、DHCPv6でグローバルアドレスを生成することも可能です。DHCPv6については第8章で説明します。

SLAACとSLAC

SLAACを規定するRFC 4862のタイトルは“IPv6 Stateless Address Autoconfiguration”です。RFC 4862が上書き廃止したRFC 2462[†17]も、タイトルはこれと同じです。これらのRFCが定義するIPv6アドレスの自動生成手法には、本書執筆時点では「SLAAC」という略称が使われています。これは、これらのRFCのタイトルにある“StateLess Address AutoConfiguration”に由来するものです。RFC 4862そのものには「SLAAC」という略称は登場しません。

興味深いのは、2005年に発行されたRFC 4192[†18]では「SLAC」という略称が使われている点です。昔はAが1つだけだったのです。

面白いことに、その後のRFCを追っていくと、いつの間にか略称にAが1つ増えているように見えます。2006年に発行されたRFC 4429[†19]では、「SLAAC」という、Aが2つ続く略称が使われています。RFC 4429以降のRFCでも同様です。

7.8 IPv6アドレスの存続期間とステート

IPv6には、IPv6アドレスが有効となる期間があらかじめ決められる仕組みがあります（リンクローカルアドレスのように、貸し出し期間が無限の場合もあります）。

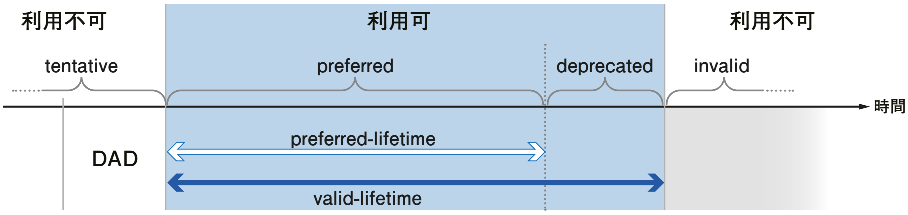
IPv6アドレスには、無効（invalid）な状態と有効（valid）な状態があります。IPv6アドレスには、そのアドレスが設定されてからの経過時間が記録されており、存続期間が切れるとそのIPv6アドレスは無効になります。無効になったIPv6アドレスは通信に利用できなくなります。

IPv6アドレスが自動生成されてからの状態を順番に並べると、次のようになります。

- tentative（仮の）
- preferred（推奨）
- deprecated（非推奨）
- invalid（無効）

IPv6アドレスが自動生成され、IPv6アドレスがネットワークインターフェースに追加されるとき、そのIPv6アドレスは「仮」の状態である**tentative**になります（図7.4）。tentativeの間、そのIPv6アドレスは無効です。tentativeのIPv6アドレスに対し、「重複するIPv6アドレ

[†17] RFC 2462：S. Thomson, T. Narten, “IPv6 Stateless Address Autoconfiguration”, 1998年12月
[†18] RFC 4192：F. Baker, E. Lear, R. Droms, “Procedures for Renumbering an IPv6 Network without a Flag Day”, 2005年9月
[†19] RFC 4429：N. Moore, “Optimistic Duplicate Address Detection (DAD) for IPv6”, 2006年4月



▶図7.4　自動設定されるアドレスの存続期間とステート

スが同一セグメント内に存在しないこと」を確認するDAD（Duplicate Address Detection）が実施されます。このDADによって、自動生成したIPv6アドレスが有効であることの確認が終わるまでは、そのIPv6アドレスはtentativeのままです。

DADが終わって重複がないことが確認されたIPv6アドレスは、有効な状態になります。有効な状態には、**preferred**と**deprecated**の2種類があります。preferredは、そのIPv6アドレスの利用が推奨される状態です。その推奨とされる期間が終わり、「利用することは非推奨でありつつもアドレスとしては有効」な状態がdeprecatedです。

deprecatedという英単語は、非推奨という意味を持ちますが、これは「完全に廃止された」という意味で使われることが多い「obsolete」よりも多少ゆるい表現といえます。実際、自動設定されたIPv6アドレスの存続期間が切れてdeprecatedの状態にあるIPv6アドレスも、使うことが禁止されているわけではありません。RFC 4862でもそのように明示されており、あくまでもそのまま使うことが「推奨されない」という状態のIPv6アドレスというわけです。

deprecatedな状態が終わると、そのIPv6アドレスは「利用が無効」である**invalid**になります。

図7.4には、**preferred-lifetime**と**valid-lifetime**という、自動設定されるIPv6アドレスの2種類の存続期間も示してあります。これはそれぞれ、IPv6アドレスが有効となる期間のうち「preferredとなる期間」と、「deprecatedでありつつもvalidであり続ける期間」を設定するためのパラメータです。preferred-lifetimeとvalid-lifetimeは、ルータなどから提供されるRouter AdvertisementメッセージやDHCPv6メッセージに含まれています。

valid-lifetimeは、「preferred-lifetimeにdeprecatedな状態を含む期間」を示します。preferred-lifetimeとvalid-lifetimeに注目すると、deprecatedは、「preferred-lifetimeは超えたけれどもvalid-lifetimeは超えていない期間」だといえます。そして、valid-lifetimeを超えてしまったIPv6アドレスが、完全に利用が禁止されたinvalidになります。

NOTE

RFC 4862の5.5.3では、Router Advertisementを受け取った際の処理として、極端に小さいvalid-lifetimeへの対処法が記されています。そこでは、valid-lifetimeの値が小さい場合には、2時間（7200秒）という値にリセットするといった内容が記述されています。

IPv6アドレスの存続期間を示すpreferred-lifetimeは、ホストがIPv6アドレスを設定後に更

[CC BY-NC-SA]

新されることもあります。たとえば、Router Advertisementメッセージをもとに IPv6アドレスの自動設定が行われている場合には、新たにRouter Advertisementメッセージを受け取った時点でpreferred-lifetimeの値がリセットされます。ステートフルDHCPv6を使っている場合には、dhcpv6 clientからrenewメッセージを送信し、それに対してdhcpv6 serverが新しいlifetimeをreplyメッセージに含めて送信することで更新されます。

7.8.1 Optimistic DAD

IPv6アドレスのステートとして、optimisticというものもあります。Optimistic DADが定義されているRFC 4429で、従来のtentative、preferred、deprecatedに加えて、IPv6アドレスの状態として新たにOptimisticが定義されています。Optimistic DADが使われる場合、IPv6アドレスが自動生成されてからの状態を順番に並べると、次のようになります。

- tentative
- optimistic
- preferred
- deprecated
- invalid

Optimistic DADは、その名が示す通り、「楽観的（optimistic）」なDADです。どうせIPv6アドレスは衝突しないだろうと楽観的に捉えて、DADの最中でも新しいIPv6アドレスが使えるようになっています。RFC 4429で定義されているoptimisticという状態は、「利用可能なIPv6アドレスではあるもののDADは完了していない」というものです。optimisticという状態の扱いはdeprecatedと同じです。

Optimistic DADではない、もともとのDADだと、DADでIPv6アドレスの重複検知が行われている間は新しいIPv6アドレスはtentativeの状態です。tentativeの状態は無効なので、そのIPv6アドレスは使えません。しかし、optimisticという状態では、IPv6アドレスは有効なので利用可能になります。これにより、Optimistic DADでは、DADが完了する前にIPv6アドレスが利用可能になります（図7.5）。たとえば、ネットワーク接続の変更後に新たなIPv6アドレスが設定される際などに、通信が切断される時間を短くするといった目的でOptimistic DADが定義されました。

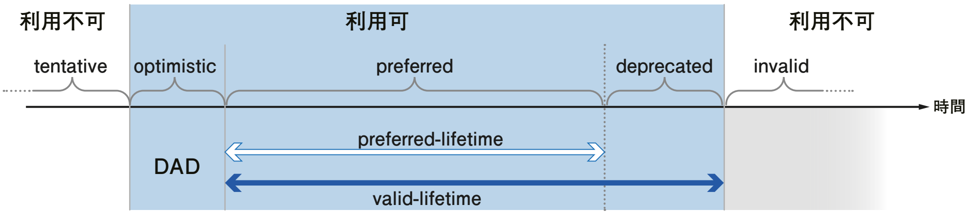
Optimistic DADは多くのシステムに実装されており、利用するかどうかを設定可能である場合も多くあります。初期設定がOptimistic DADを利用するようになっているかどうかはシステム依存です。

7.8.2 preferredとdeprecatedの活用法

IPv6アドレスが有効な状態として、有効であり利用が推奨されるpreferredと、有効ではあるが利用が推奨されていないdeprecatedの2種類が考慮されるのはなぜでしょうか。preferredとdeprecatedの関係がポイントになるのは、送信元IPv6アドレス選択にあるといえます。

RFC 4862では、deprecatedの状態にあるIPv6アドレスを、新規の通信で送信元IPv6アド

CC BY NC SA



▶図7.5　自動設定されるアドレスの存続期間とステート（Optimistic DAD）

レスとして利用することは避けるべきであるとしています。つまり、送信元IPv6アドレスとしてはpreferredの状態にあるIPv6アドレスを使うべきとしています。デフォルトIPv6アドレス選択（RFC 6724[†20]）においても、「deprecatedなIPv6アドレスよりも、preferredなIPv6アドレスを送信元IPv6アドレスとして選択する」というルールがあります。

その一方でRFC 4862では、「すでに継続しているTCP接続などではdeprecatedの状態にあるアドレスを使い続けてもよい」としています。これはつまり、deprecatedの仕様は「既に存在している通信では継続して使ってもよいが、新たな通信では避けるべき」だと言えるでしょう。実際、ある特定のネットワークインターフェースに複数のIPv6アドレスが設定されている場合、deprecatedなものよりも、preferredなIPv6アドレスの方が送信元IPv6アドレスとして利用されやすくなります[†21]。

deprecatedなIPv6アドレスよりも、preferredなIPv6アドレスが送信元IPv6アドレスとして選択されやすくなるという仕組みの活用方法は、いくつかのRFCで言及されています。たとえばRFC 4862では、これらの仕組みがIPv6でのサイトリナンバリングで有用となる可能性が示唆されています。IPv6アドレスのlifetimeという要素を調整することで、個々の通信セッションにおいて古いIPv6アドレスプレフィックスの利用を減らし、新しいIPv6アドレスの利用を促すという考え方です。

ただし、RFC 4862にはIPv6でのサイトリナンバリングに関する詳細は記述されてません。IPv6でのサイトリナンバリングに関しては他のRFCで考察されています。本書では13.3節でサイトリナンバリングに関して改めて説明しています。

サイトリナンバリングよりも一般に身近かもしれない活用方法としては、RFC 8981（旧RFC 4941）のTemporary IPv6アドレスがあります。Temporary IPv6アドレスは、その名のとおり一時的なものであり、一定の期間が過ぎると使われなくなることが想定されています。Temporary IPv6アドレスがdeprecatedになった時点で新しいTemporary IPv6アドレスが生成され、新たに開始される通信ではdeprecatedなものではなく、preferredな状態にあるTemporary IPv6アドレスの利用が促されるようになります。

†20　RFC 6724：D. Thaler, R. Draves, A. Matsumoto, T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", 2012年9月

†21　ただし、送信元IPv6アドレスの選択は宛先に依存しますし、さまざまな要素が絡むので、どんな場合でも必ずそうなるとは限りません。

CC BY NC SA

7.9 Router Advertisement メッセージによる DNS 情報の配送

Router Advertisement メッセージでは、2000 年頃まで、DNS 情報を送れない仕様になっていました。IPv6 が設計された当初は、Router Advertisement メッセージに DNS 情報を付加できず、DNS 情報は DHCPv6 を利用して別途配布することになっていました。そのような設計になっていたのは、ネットワーク層の機能である Router Advertisement メッセージを利用して上位層で利用する DNS 情報を配送することはレイヤーバイオレーションであり、プロトコルとして美しくないという理由によるものです。

Router Advertisement メッセージのオプションとして DNS 情報を付加する仕様は、2007 年に、RFC 5006[†22] として発行されました。RFC 5006 は実験的（Experimental）とされていましたが、2010 年には RFC 5006 を上書き廃止する形で、RFC 6106 が標準（Proposed Standard）として発行されました。RFC 5006 で定義されていたのはキャッシュ DNS サーバを示す RDNSS（Recursive DNS Server）オプションのみですが、RFC 6106 では DNS suffix search list を示す DNSSL（DNS Search List）オプションも定義されています。RDNSS オプションについては 6.5.6 項、DNSSL オプションについては 6.5.7 項を参照してください。

2017 年 3 月には RFC 6106 の後継となる RFC 8106 が発行されています。RFC 8106 と RFC 6106 の違いとしては、下記のような点が挙げられます。

- RDNSS オプションや DNSSL オプションの Lifetime の最大値として RFC 6106 よりも大きな値が使われるようになった
- Neighbor Solicitation メッセージを送信して RDNSS オプションや DNSSL オプションで示された内容を確認することへの言及が削除された
- RDNSS としてリンクローカルアドレスを利用可能になった
- RDNSS や DNS search domain としてホスト側が採用する値を 3 つとする推奨が削除された
- 実装上の注意点が一部削除された

†22 RFC 5006：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Option for DNS Configuration”, 2007 年 9 月

CC BY NC SA

[書籍版では空白のページです]

CC BY NC SA

第8章

DHCPv6

IPv6では、IPv4におけるDHCPと似た**DHCPv6**というプロトコルが利用されます。DHCPv6はRFC 8415[†1]で定義されています。

DHCPv6には、大きく分けて次の3種類の利用形態があります。

- ステートレスDHCPv6
- ステートフルDHCPv6
- DHCPv6-PD

上記のうち、IPv4におけるDHCPの利用形態に近いのは、2つめに挙げた**ステートフルDHCPv6**です。

一方、**ステートレスDHCPv6**では、IPv6アドレス以外のパラメータについてはDHCPv6サーバから受け取り、IPv6アドレス設定に関しては別の方法で行われます。たとえば、SLAAC（第7章参照）を利用したIPv6アドレスの自動設定を行います[†2]。

このように、クライアントのIPv6アドレスをDHCPv6サーバによって管理するのがステートフルDHCPv6、管理しないのがステートレスDHCPv6といえるでしょう。

3つめの**DHCPv6-PD**は、ネットワークプレフィックスをクライアントに対して委任するための仕組みです。ISPなどが運用するDHCPv6-PDルータが、顧客側のブロードバンドルータ[†3]などで運用されているDHCPv6-PDルータに対して、IPv6プレフィックスを委任します。DHCPv6-PDは、ルータがルータに対して設定情報を提供するというのが大きな特徴です。IPv4では、DHCPを規定しているRFC 2131[†4]のSection 1.3に、“DHCP is not intended for use in configuring routers.”という表現があります。つまり、IPv4のDHCPでは、ルータをクライアントとすることが想定されていないのです。ルータをクライアントとすることを想定しているDHCPv6-PDの存在は、DHCPv6がIPv4のDHCPと大きく異なる点のひとつであるとい

†1 RFC 8415：T. Mrugalski,M. Siodelski,B. Volz,A. Yourtchenko,M. Richardson,S. Jiang,T. Lemon,T. Winters, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2018年11月

†2 ステートレスDHCPv6でIPv6アドレス以外のパラメータを受け取りつつ、手動設定で行うこともできます。

†3 こうした機器をCPE（Customer Premises Equipment）と呼ぶことがあります。

†4 RFC 2131：R. Droms, “Dynamic Host Configuration Protocol”, 1997年3月

CC BY NC SA

えます。

DNSサーバの情報をDHCPv6で取得する必要はあるか

ステートレスDHCPv6では、DNSサーバの情報をDHCPv6サーバから取得させます。一方、SLAACではDNSサーバに関する情報をRouter AdvertisementメッセージのRDNSSオプションを使って取得できます。

DNSサーバの情報をIPv6ホストに提供するという点だけ見ると、SLAACだけで完結できそうですが、DHCPv6を使うことにも利点はあります。たとえば、ステートレスDHCPv6であれば、NTP（RFC 5908[†5]）、SIP（RFC 3319[†6]）、NIS（RFC 3898[†7]）などの情報も配信可能になります。

ホストにおけるDNSサーバに関する情報の取得のためにDHCPv6を使うかどうかについては、RFC 4339[†8]で考察されています。RFC 4339が発行されたのは2006年なので、情報が少し古い部分もありますが、DHCPv6を使う場合とSLAACのみ（DHCPv6を使わない）で行う場合の違いがわかりやすくまとめられています。

DNSサーバの情報取得に関するDHCPv6の必要性については8.5.3項でも改めて説明します。

8.1　IPv4のDHCPとDHCPv6の違い

DHCPとDHCPv6は、設計思想や歴史的経緯からしてまったく異なるプロトコルであり、それぞれの仕様には前提となるIPv4とIPv6の違いが明確に現れています。IPv4用のDHCPと、IPv6用のDHCPv6の主な違いとして、以下のような点が挙げられます。

- DHCPクライアントは`255.255.255.255`というブロードキャストアドレスに対して`discover`メッセージを送信しますが、DHCPv6クライアントは`ff02::1:2`というマルチキャストアドレスに対して`solicit`メッセージを送信する
- DHCPはBOOTPを拡張したものであり、BOOTPのために割り当てられたUDPポート番号67と68を利用する。DHCPv6は、DHCPv6のために割り当てられたUDPポート番号546と547を利用する
- DHCPv6を利用してIPv4アドレスを割り当てることもできる
- DHCPv6にはステートレスとステートフルの2通りの使い方がある
- クライアントがIPアドレスの割り当てをサーバに対して要求する際、DHCPではIPv4アドレスが決定していないので、クライアントからのIPv4パケットでは送信元アドレスとして`0.0.0.0`を設定する。DHCPv6では、送信元アドレスとしてリンクローカルアドレ

†5　RFC 5908：R. Gayraud, B. Lourdelet, “Network Time Protocol (NTP) Server Option for DHCPv6”, 2010年6月

†6　RFC 3319：H. Schulzrinne, B. Volz, “Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers”, 2003年7月

†7　RFC 3898：V. Kalusivalingam, “Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2004年10月

†8　RFC 4339：J. Jeong, “IPv6 Host Configuration of DNS Server Information Approaches”, 2006年2月

CC BY NC SA

スを使う

- DHCPでは、サーバからクライアントへの応答に、`255.255.255.255`というブロードキャストアドレスを使う。DHCPv6では、サーバからクライアントへの応答を、クライアントのリンクローカルアドレスへのユニキャストとして送信する
- DHCPは、ホストに対してIPアドレスしか設定できない。DHCPv6では、ネットワークプレフィックスを割り当てるためのDHCPv6-PDがある

8.2 IPv4のDHCP

DHCPv6の説明をする前に、IPv4のDHCPについて紹介しておきます。DHCPが登場する以前は、ユーザが各自で通信に必要な情報を取得し、自分で値を設定していました。設定する値はネットワークごとに異なるので、一概には決められません。もしIPアドレスが重複するようなことがあれば、ネットワークに障害が発生するおそれもあります。そのため、利用可能な空いているIPアドレスを、個々のユーザが常に把握できるようにしておかなくてはなりませんでした。

DHCPで自動的にIPアドレスその他を割り当てることにより、インターネットに接続するための負荷が大幅に軽減されました。DHCPを利用して自動的に設定できる主な情報としては、以下のようなものが挙げられます（これで全部ではありません）。

- IPv4アドレス
- ネットマスク
- ブロードキャストアドレス
- デフォルトルータ
- キャッシュDNSサーバ

NOTE

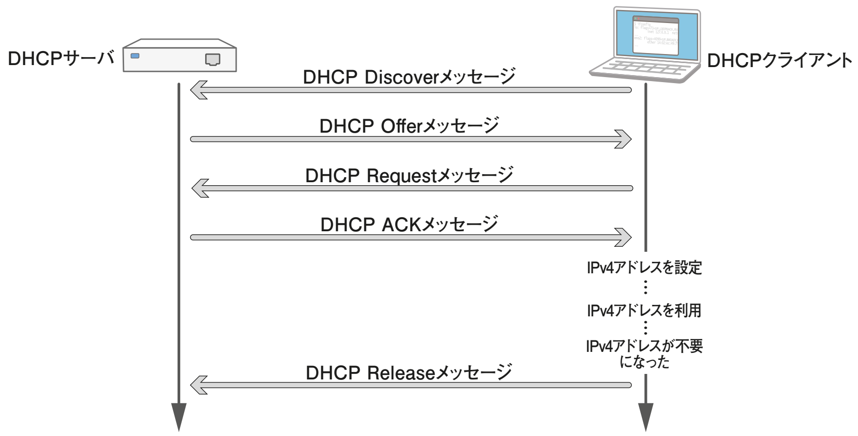
IPv6で使われるDHCPv6を知るために、IPv4のDHCPを必ずしも理解している必要はありませんが、IPv4とIPv6のデュアルスタック環境を運用する際にIPv4のDHCPに関する知識も必要になるので、本節では概略を説明しています。詳細についてはRFC 2131を参照してください。

8.2.1 DHCPでのIPv4アドレス取得時の動作

DHCPを使ってDHCPクライアントがネットワーク接続に必要な情報を取得して設定するまでの流れを図8.1に示します。DHCPサーバは、あるセグメントで利用可能なIPアドレスや接続に必要な各種情報を知っているサーバです。DHCPクライアントは、そのネットワークのことは何も知りませんが、インターネットに接続したいので、DHCPプロトコルを利用してDHCPサーバから各種情報を得ようとします。

DHCPクライアントは何もわからないので、「DHCPサーバがいるのであれば情報をください」という意味のメッセージ（DHCP Discoverメッセージ）をネットワークに送付します。このときの宛先アドレスは、同一セグメント上のすべてのノードを示すブロードキャストアドレ

CC BY NC SA



▶図8.1　DHCPクライアントがDHCPサーバからネットワーク接続に必要な情報を取得する流れ

ス255.255.255.255です。

DHCPクライアントからのDHCP Discoverメッセージを受け取ったDHCPサーバは、「こんな設定が可能です」という意味のメッセージ（DHCP Offerメッセージ）を、DHCPクライアント宛のユニキャストで送信します。DHCP Offerを受けたDHCPクライアントは、DHCPサーバから受け取った設定情報が気に入れば「これを使わせてください」という意味のメッセージ（DHCP Requestメッセージ）をDHCPサーバに送信します。

DHCPサーバは、DHCPクライアントからのDHCP Requestを了承し、DHCP ACKを返します。DHCPクライアントがDHCP Offerを受け取ってすぐに設定しないのは、複数のDHCPサーバが存在している状態で1つのDHCP Discoverに対して2つ以上のDHCP Offerがあったときに、DHCPクライアントがどれか1つを選んでDHCP Requestを送付できるようにするためです。

8.2.2　IPv4アドレス解放時の動作

DHCPサーバは、そのセグメントで現在空いているIPアドレスを管理してDHCPクライアントに配っているので、あるDHCPクライアントがIPv4アドレスを使わなくなったときには、そのことを教えてもらったほうが効率良くIPv4アドレスを活用できます。そのため、電源が切られるなどの理由でIPv4アドレスが不要になったDHCPクライアントは、DHCPサーバから与えられたIPv4アドレスを解放するためにDHCP Releaseメッセージを送付します。DHCPサーバは、DHCP Releaseメッセージを受け取ると、そのIPv4アドレスが解放されたものとして扱い、別の要求を受けたときに使えるようにします。

もし、DHCPクライアントがDHCP Releaseメッセージを送付せずにネットワークから離脱した場合には、DHCPサーバ側でタイムアウトします。とはいえ、たとえばサスペンド状態で長時間ダウンしていたノートPCが、タイムアウト後に急にネットワークに再接続された結果、古いIPv4アドレスを使ってしまってIPv4アドレスが衝突してしまう、ということも割と頻繁に発生します。

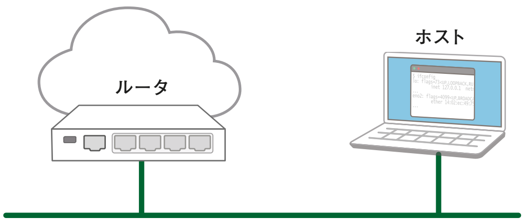
CC BY NC SA

8.3　DHCPv6の概要

DHCPv6について大まかなイメージを把握するために、小規模なネットワークにおけるステートレスおよびステートフルのDHCPv6での動作例を紹介します。IPv6で運用されているルータとホストが同一リンクに1台ずつ接続されている図8.2のようなネットワーク（7.2節で紹介したSLAACでの動作例と同じ）で、ホストが起動と同時にIPv6アドレスの自動設定を行う場合を考えます。ルータにはDHCPv6サーバの機能が実装されており、ホストではDHCPv6クライアントが稼働しているものとします。

NOTE

ここで紹介するのは、あくまでも考えられる利用例のひとつです。アドレス自動設定に関連する各RFCには、具体的な手順やオプションの組み合わせ例などは記載されておらず、開発者や運用者が柔軟に実装できるようになっています。本書では、あくまでもDHCPv6の全体像を説明することを目的に、考えられる手順とオプションの組み合わせ例を紹介しています。



▶ 図8.2　小規模なネットワーク例

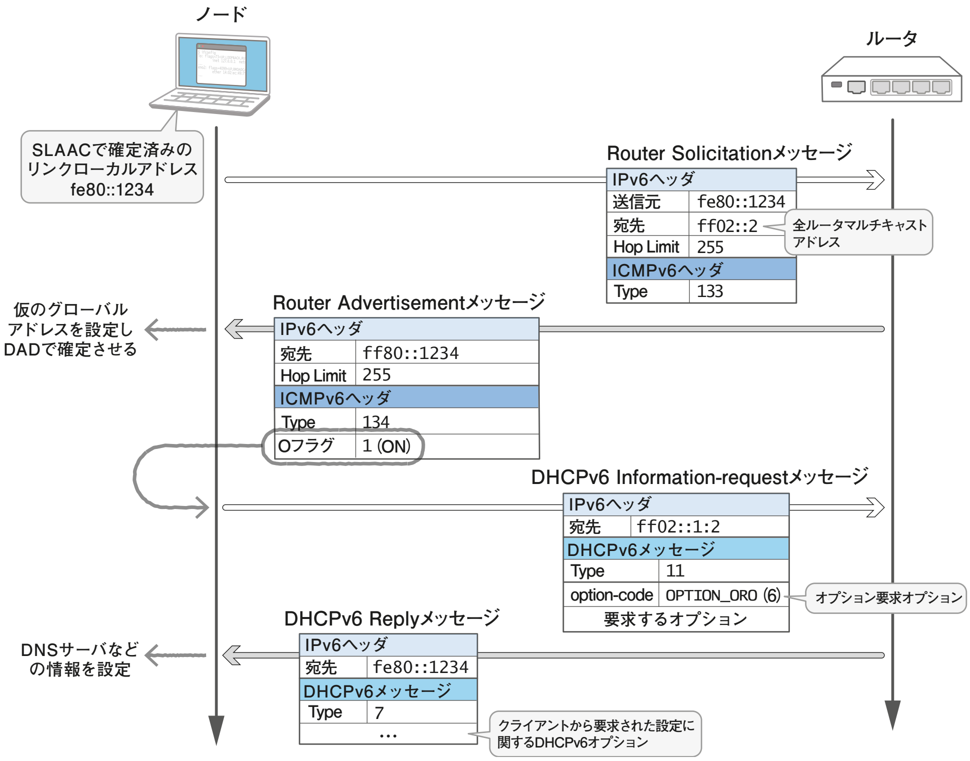
IPv4におけるDHCPとDHCPv6の大きな違いとして、最初にルータがIPアドレスの自動設定に関与するという点が挙げられます。具体的には、DHCPv6が使われることを示すために、ルータからのRouter Advertisementメッセージに含まれるOフラグ（Other configuration）やMフラグ（Managed address configuration）のビットを1にすることが可能です。ただし、OフラグやMフラグはDHCPv6が利用可能であることを示すものであり、DHCPv6を運用するうえで必須というわけではありません。これらのフラグが0という状況であっても、DHCPv6の利用が禁止されているわけではないのです。

8.3.1　ステートレスDHCPv6利用の流れ

ステートレスDHCPv6でホストのIPv6アドレスを設定するまでの流れを図8.3に示します。ステートレスDHCPv6では、IPアドレスの設定はDHCPv6で行わず、Router Advertisementメッセージに含まれる情報をもとにホストが自動生成します。ステートレスDHCPv6については8.5節で改めて解説します。

ステートレスDHCPv6が使われるとき、ルータからのRouter AdvertisementメッセージのOフラグを1に設定することで、DHCPv6サーバによる情報提供が可能であることを示すことができます。ステートレスDHCPv6を利用してDNSやNTPなどに関連する情報を得る場合、ホストはDHCPv6 Information-requestメッセージをリンクに送信します。DHCPv6

CC BY NC SA



▶ 図8.3　小規模なネットワーク例におけるステートレスDHCPv6の一例

Information-requestメッセージには、DHCPv6クライアントが要求する設定情報を伝えるためのオプションが含まれています。DHCPv6サーバは、Information-requestメッセージへの返答として、必要な設定情報を含むDHCPv6 Replyメッセージを送信します。DHCPv6クライアントは、DHCPv6 Replyメッセージに含まれる情報をもとに設定を行います。

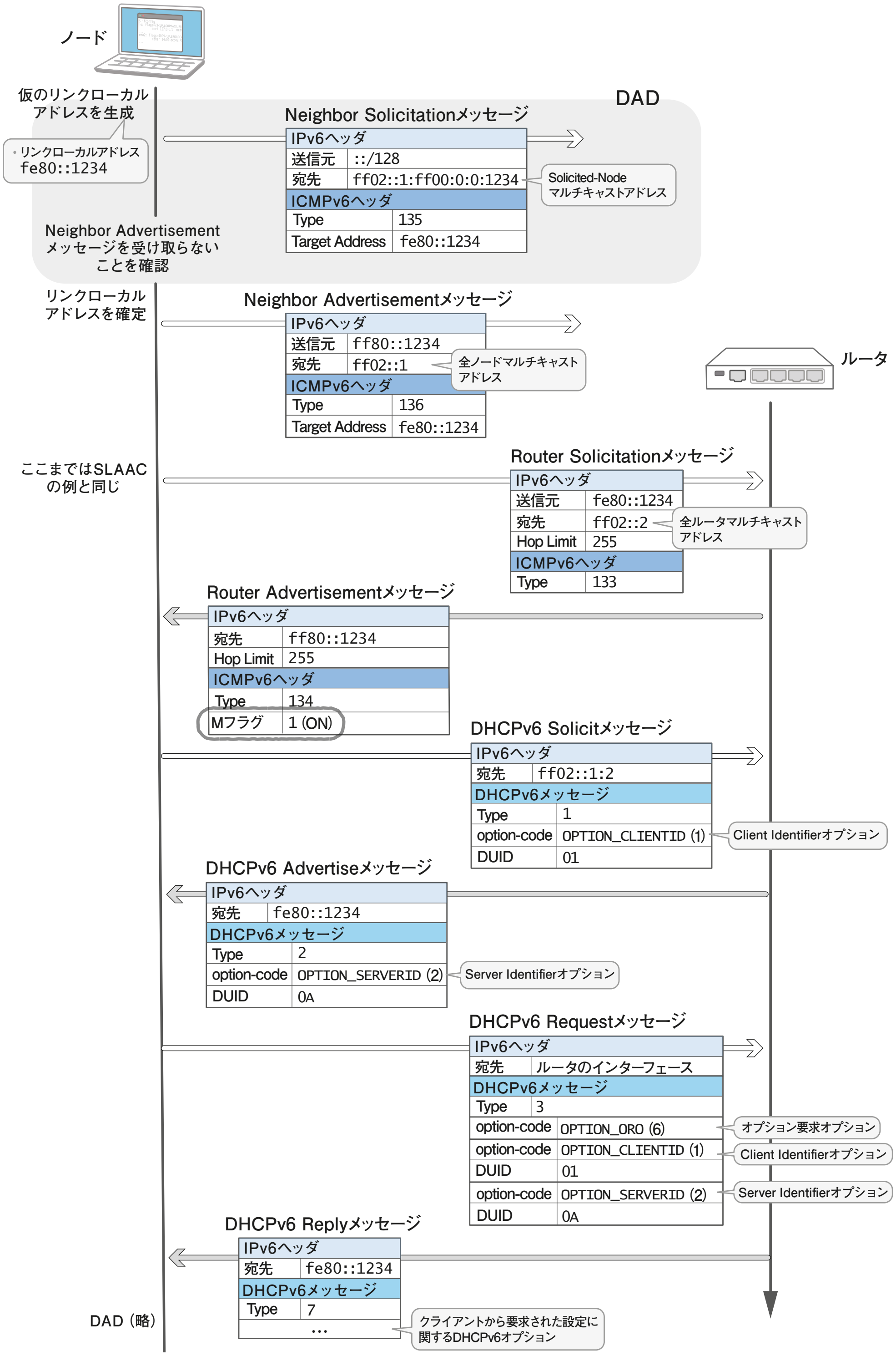
8.3.2　ステートフルDHCPv6利用の流れ

ステートフルDHCPv6でDNSサーバに関する情報をホストで設定するまでの流れを図8.4に示します。

ステートフルDHCPv6が使われるとき、ルータからのRouter AdvertisementメッセージのMフラグを1に設定することができます。このとき、ホストはDHCPv6 Solicitメッセージをリンクに送信してDHCPv6サーバを探します。DHCPv6サーバがAdvertiseメッセージをユニキャストでDHCPv6クライアントへと返答し、DHCPv6クライアントはDHCPv6サーバに対してRequestメッセージを使ってIPv6アドレスやネットワークに付随する情報を要求します。DHCPv6サーバは、Replyメッセージによって各種設定情報をDHCPv6クライアントへと伝え、DHCPv6クライアントはDHCPv6サーバからの情報をもとに設定を行います。

ステートフルDHCPv6については8.6節で改めて解説します。

CC BY NC SA



▶ 図 8.4　小規模なネットワーク例におけるステートフル DHCPv6 の一例

CC BY NC SA

8.3.3　DHCPv6プロトコルで使うポートとアドレス

DHCPv6クライアントは、UDPポート番号546で待ち受けます。DHCPv6サーバは、UDPポート番号547で待ち受けます。

また、以下の2種類のマルチキャストアドレスがDHCPv6で利用するためにIANAで予約されています。

- All_DHCP_Relay_Agents_and_Serversアドレス（ff02::1:2）
  クライアントがサーバもしくはリレーエージェントと通信するために利用するリンクスコープマルチキャストアドレスです。DHCPv6サーバとリレーエージェントは、このマルチキャストアドレスに参加します。
- All_DHCP_Serversアドレス（ff05::1:3）
  リレーエージェントがサーバと通信するために利用するサイトスコープマルチキャストアドレスです。DHCPv6サーバは、このマルチキャストアドレスに参加します。

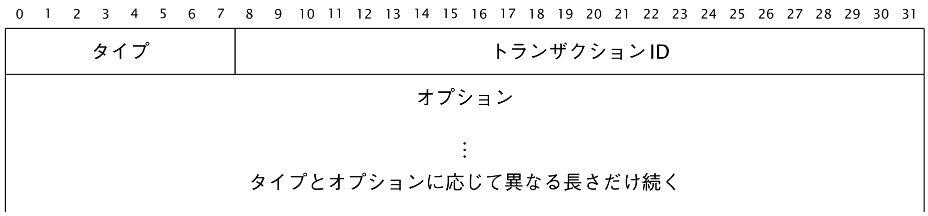
8.3.4　DHCPv6のメッセージタイプ

RFC 8415で定義されているDHCPv6メッセージを表8.1に示します。

DHCPv6のメッセージタイプは、RFC 8415以外のRFCでも定義されています。DHCPv6メッセージタイプやオプションコードなどの一覧は、IANAのサイトで公開されています[†9]。

8.3.5　DHCPv6メッセージの基本フォーマット

すべてのDHCPv6メッセージは、図8.5に示す基本フォーマットを持ちます。



▶ 図8.5　DHCPv6メッセージの基本フォーマット

先頭1オクテットがメッセージのタイプ（表8.1参照）を示します。続く3オクテットが、要求と応答の対応を示すトランザクションIDです。

トランザクションIDの後には、いくつかのオプションが連続して続きます。各オプションの境界を2オクテットや4オクテットにそろえるためのパディングは必要ありません。

---

†9　DHCPv6パラメータ一覧：
https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml

▶ 表 8.1 DHCPv6 メッセージのタイプ

番号	メッセージ名	内容
0	Reserved	未定義
1	Solicit	サーバを探索するためにクライアントが送信するメッセージ
2	Advertise	クライアントからの Solicit メッセージに対して DHCP サービスが可能であることを伝えるために、サーバから送信されるメッセージ
3	Request	クライアントからサーバに対し、IPv6 アドレスを含む設定情報を特定のサーバから受け取るために送信するメッセージ
4	Confirm	過去に割り当てされた IPv6 アドレスが利用可能であるかをクライアントが確認するためのメッセージ
5	Renew	クライアントからサーバに対し、すでに割り当てられた設定パラメータの有効期間を延ばすために送信されるメッセージ
6	Rebind	クライアントが、すでに割り当てが行われている設定パラメータの有効期間を延ばすために、任意のサーバに対して送信するメッセージ。Renew メッセージに対する応答を受け取れない場合に送信される
7	Reply	クライアントからの Solicit、Request、Renew、Rebind の各メッセージに対し、サーバが割り当てた IPv6 アドレスと設定情報を返信するためのメッセージ。クライアントからの Information-request メッセージへの返答としても、設定情報を含む Reply メッセージを送信する。Confirm メッセージへの返答としても、Reply メッセージを送信する。Release もしくは Decline メッセージへの受信確認としても Reply メッセージを送信する
8	Release	クライアントが、サーバから割り当てられた IPv6 アドレスを利用しなくなったことをサーバに伝えるために送信するメッセージ
9	Decline	サーバから割り当てられた IPv6 アドレスがすでに使われていることをクライアントからサーバに伝えるためのメッセージ
10	Reconfigure	新しい設定情報をサーバからクライアントに対して伝えるために利用されるメッセージ
11	Information-request	IPv6 アドレスを設定せずに設定情報のみをクライアントが要求するために利用されるメッセージ。ステートレス DHCPv6 で利用される
12	Relay-Forward	サーバに対してリレーエージェントがメッセージを伝えるために使われるメッセージ
13	Relay-Reply	リレーエージェントを通じてサーバがクライアントに対してメッセージを伝えるために使われるメッセージ

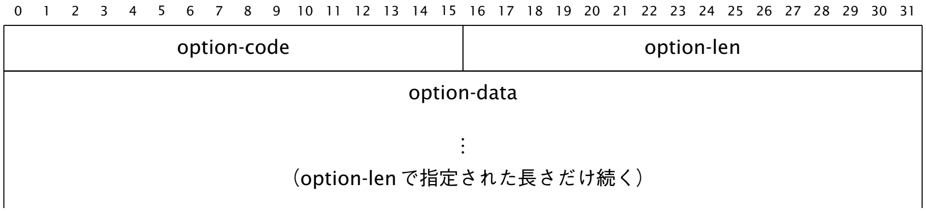
含まれるオプションの種類は、DHCPv6 メッセージのタイプによって異なります。DHCPv6 のオプション一覧は、IANA のサイトにまとめられています[†10]。以降では、DHCPv6 メッセージの代表的なオプションを中心に説明します。

[†10] DHCPv6 オプション一覧：
https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml#dhcpv6-parameters-2

CC BY NC SA

8.3.6　DHCPv6メッセージのオプション

DHCPv6メッセージのオプションは、図8.6のような基本フォーマットを持ちます。



▶ 図8.6　DHCPv6メッセージオプションの基本フォーマット

DHCPv6メッセージのオプションは、4.3.3項で説明したTLV形式として定義されています。具体的には、オプションの先頭16ビットがオプションの種類を示すoption-code、続く16ビットがoption-dataフィールドの長さを示すoption-len、その後ろが、オプションによって異なる形式のデータoption-dataです。

NOTE

DHCPv6メッセージで利用するオプションを新たに定義する人のための注意点などがRFC 7227[†11]としてまとめられています。このRFCでは、どのようなときにDHCPv6を利用すべきかや、DHCPv6を利用してクライアントに提供するデータの内容に応じたオプションフォーマットの例、実際にその型を利用したフォーマットを採用しているRFCなどが紹介されています。DHCPv6メッセージのオプションについてさらに詳しく知りたい方はRFC 7227も参照してください。

以下、いくつかのDHCPv6メッセージオプションについて、option-codeとoption-dataの内容をまとめます。

8.3.7　ステータスコードオプション（OPTION_STATUS_CODE）

DHCPv6メッセージのステータスを示すためのオプションです。他のオプションの内部に含まれることもあります。RFC 8415では表8.2に示す7種類のステータスが定義されています。

DHCPv6のためのステータスコードは、他のRFCでも定義されています。IANAのサイト[†12]で一覧を参照できます。

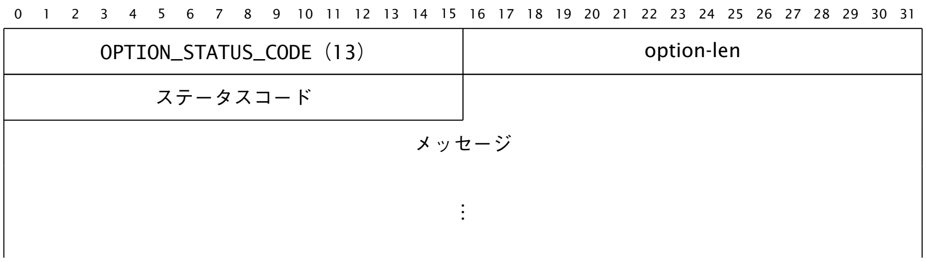
図8.7に、ステータスコードオプションのフォーマットを示します。ステータスコードオプションのoption-codeは`OPTION_STATUS_CODE`（13）です。2オクテットのステータスコードに続いて、ステータスに関するメッセージをUTF-8で記述できます。

[†11] RFC 7227：D. Hankins,T. Mrugalski,M. Siodelski,S. Jiang,S. Krishnan, “Guidelines for Creating New DHCPv6 Options”, 2014年5月

[†12] DHCPv6ステータスコード一覧：
https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml#dhcpv6-parameters-5

▶ 表 8.2　DHCPv6 メッセージのステータスコード

ステータス	コード	内容
Success	0	成功を表す
UnspecFail	1	詳細は明示せずに失敗を表す。他のステータスコードが当てはまらない場合に、クライアントもしくはサーバから送信される
NoAddrsAvail	2	IA（8.6.1 項参照）に対して提供できるアドレスがサーバに存在しないことを示す
NoBinding	3	クライアントに関する記録がサーバに存在しないことを示す
NotOnLink	4	アドレスに使われているプレフィックスが、接続されたリンクに対して適切ではないことを示す
UseMulticast	5	サーバがクライアントに対して、All_DHCP_Relay_Agents_and_Servers を使うことを促すために使われる
NoPrefixAvail	6	サーバが IA_PD に対してアサインできるプレフィックスが存在しない

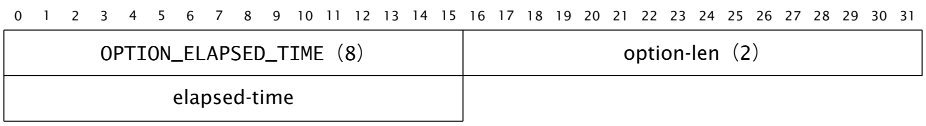


▶ 図 8.7　ステータスコードオプション

8.3.8　経過時間オプション（OPTION_ELAPSED_TIME）

クライアントからのメッセージには、必ず、クライアントが DHCP トランザクションを開始してからの経過時間を伝えるオプションを含める必要があります。RFC 8415 では、このフィールドの値が大きい場合に正規のサーバが応答していないとみなして予備のサーバが応答する例が紹介されています。

図 8.8 に、経過時間オプションのフォーマットを示します。経過時間オプションの option-code は OPTION_ELAPSED_TIME（8）です。経過時間は 1/100 秒単位で示されます。elapsed-time フィールドの値は 16 ビットの符号なし整数（0 および正の整数）として表現されます（65535 を超える値はすべて 0xffff とします）。したがって、option-len の値は 2 です。なお、最初のメッセージでは経過時間を 0 に設定します。



▶ 図 8.8　経過時間オプション

8.3.9　オプション要求オプション（OPTION_ORO）

必要な情報を相手に対して要求するメッセージで、相手に要求するDHCPv6オプションを個別に指定したい場合には、オプション要求オプションが利用できます。たとえば、ステートレスDHCPv6でクライアントがDNSサーバに関する情報を取得するため、DNS Recursive Name Serversオプションと DNS Search Listオプションを指定したオプション要求オプションを含めてInformation-requestメッセージを送信する、といった使い方をします。また、サーバのほうからクライアントが要求すべき項目を示すため、サーバからクライアントへのReconfigureメッセージにオプション要求オプションを含めることもあります。

図8.9に、オプション要求オプションのフォーマットを示します。オプション要求オプションのoption-codeはOPTION_ORO（6）です。要求するオプションは、オプションコードの配列として表現します。オプションコードは2オクテットで表されるので、option-lenフィールドは、含まれるオプションコードの数に2をかけた値となります。たとえば、DNS Recursive Name Serversオプション（オプションコード23）とDNS Search Listオプション（オプションコード24）を指定する場合、option-lenは4で、それに続くオプションコードの配列は23（0x0017）と24（0x0018）となります。

0–15	16–31
OPTION_ORO（6）	option-len（オプション数×2）
1つめのオプション	2つめのオプション
…	

▶ 図8.9　オプション要求オプション

8.3.10　Client Identifierオプション（OPTION_CLIENTID）とServer Identifierオプション（OPTION_SERVERID）

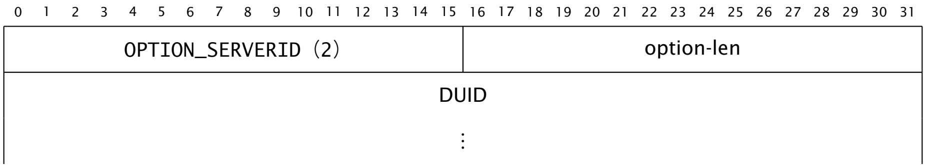
DHCPv6では、クライアントとサーバを識別するためのオプションとして、Client IdentifierオプションとServer Identifierオプションがあります。それぞれ、DHCPv6で通信相手を識別するために利用するDUIDという値を1つ含むフォーマットです。DUIDについては8.4節で説明します。

クライアントを識別するClient Identifierオプションのフォーマットを図8.10に示します。Client Identifierオプションのoption-codeはOPTION_CLIENTID（1）です。

0–15	16–31
OPTION_CLIENTID（1）	option-len
DUID ⋮	

▶ 図8.10　Client Identifierオプション

サーバを識別するServer Identifierオプションのフォーマットを図8.11に示します。Server Identifierオプションのoption-codeは`OPTION_SERVERID`（2）です。



▶ 図8.11　Server Identifierオプション

8.4 DUID

DHCPv6では、DHCPクライアントとDHCPサーバが、それぞれ個別の識別子としてDUID（DHCP Unique Identifier）を持ちます。DUIDは、クライアントやサーバが通信相手を識別するために利用し、Client IdentifierオプションとServer Identifierオプションの中に含まれるデータとして使われます。

たとえばDHCPサーバでは、DHCPクライアントから渡されたDUIDを利用して、そのDHCPクライアントに送付する設定パラメータを選択します。DHCPサーバのDUIDは、DHCPクライアント側でDHCPサーバを識別する必要がある場合に利用されます。

DUIDには、サーバやクライアントの識別に利用する値によって、いくつかタイプがあります。2021年9月時点では、DUIDにはRFC 8415で定義されているDUID-LLT、DUID-EN、DUID-LLと、RFC 6355[†13]で定義されているDUID-UUIDの、計4種類が利用できます。

8.4.1 DUID-LLT

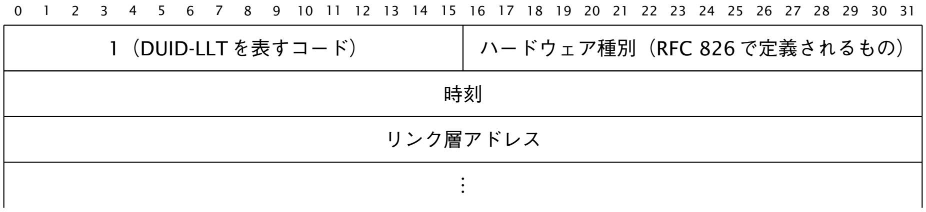
DUID-LLTは、DUID Based on Link-layer Address Plus Timeを意味し、リンク層アドレスと時刻を組み合わせたDUIDです。時刻は、2000年1月1日0時0分0秒（UTC）からの秒数を32ビットで表したものです。

RFC 8415では、複数のネットワークインターフェースが存在する場合であっても、同一機器ではどれか1つのネットワークインターフェースを選択して1つのDUID-LLTを利用することを推奨しています[†14]。また、一度生成されたDUID-LLTは保持し続けることが推奨されています。具体的には、DUID-LLTを生成するためにリンク層アドレスを利用したネットワークインターフェースが取り除かれたとしても、同一の機器ではいままでと同じDUID-LLTを使い続けることが推奨されています。DUID-LLTはネットワークインターフェースのリンク層アドレスだけで決まらず、時刻を組み合わせているので、ネットワークインターフェースが別の機器に移されたとしてもDUID-LLTとしては衝突が発生しにくい設計になっています。

†13 RFC 6355：T. Narten, J. Johnson, “Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)”, 2011年8月

†14 RFC 8415のSection 11.2には次のように記述されています。“The choice of network interface can be completely arbitrary, as long as that interface provides a globally unique link-layer address for the link type; the same DUID-LLT SHOULD be used in configuring all network interfaces connected to the device, regardless of which interface’s link-layer address was used to generate the DUID-LLT.”

DUID-LLTのフォーマットを図8.12に示します。



▶ 図8.12　DUID-LLT

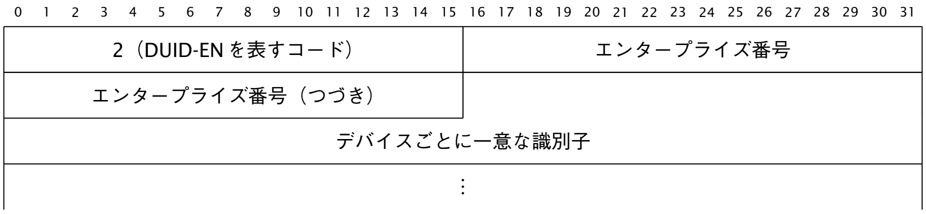
8.4.2　DUID-EN

DUID-ENは、DUID Assigned by Vendor Based on Enterprise Numberを意味し、ベンダーを表す32ビットのエンタープライズ番号と、デバイスごとに一意な識別子を組み合わせたDUIDです。

32ビットのエンタープライズ番号は、IANAによって割り当てられたエンタープライズ番号[†15]を利用します。

デバイスごとに一意な識別子は、ベンダーが機器を出荷する段階で消去不可能なストレージに対して設定した値を使うことが想定されています。

DUID-ENのフォーマットを図8.13に示します。



▶ 図8.13　DUID-EN

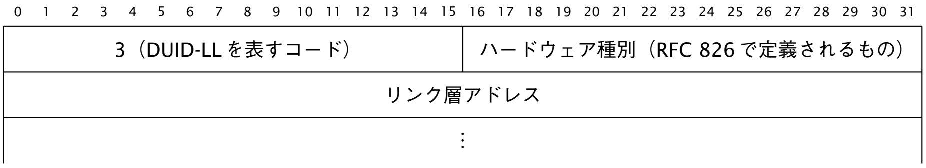
8.4.3　DUID-LL

DUID-LLは、DUID Based on Link-layer Addressを意味し、リンク層アドレスを利用した識別子です。

DUID-LLは、取り外しできないネットワークインターフェースが付いている機器での利用を想定しています。ネットワークインターフェースが変わる可能性がある機器がDUID-LLを利用することは禁止されています。

DUID-LLのフォーマットを図8.14に示します。

[†15] PRIVATE ENTERPRISE NUMBERS：https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers

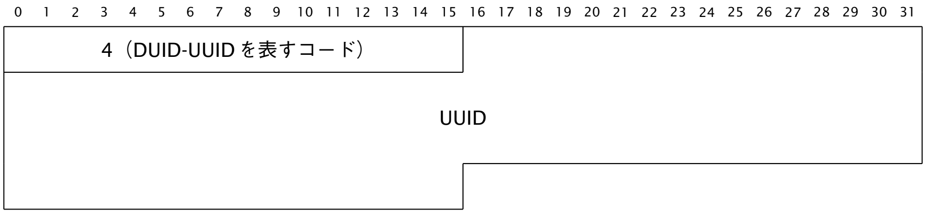


▶ 図 8.14　DUID-LL

8.4.4　DUID-UUID

DUID-UUID では、RFC 4122[†16]で定義されている 128 ビットの UUID を利用します。たとえば、x86 系の CPU を搭載した機器では内蔵された UUID を利用できます。

DUID-UUID のフォーマットを図 8.15 に示します。DUID-UUID であることを示す 4 に続いて、128 ビットの UUID が続きます。



▶ 図 8.15　DUID-UUID

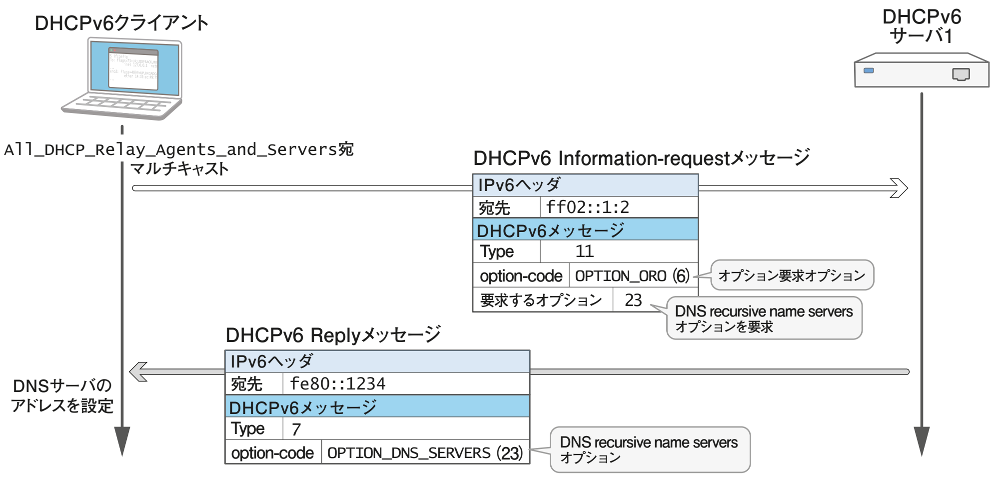
8.5　ステートレス DHCPv6

ステートレス DHCPv6 は、クライアントサーバ間の 2 通のメッセージで完結するシンプルな方式です。IPv6 アドレス設定を必要とせず、DNS などの情報のみが必要なクライアントのために、RFC 8415 で定義されています。

ステートレス DHCPv6 では、図 8.16 のように、まず DHCPv6 クライアントが Information-request メッセージをマルチキャストで `All_DHCP_Relay_Agents_and_Servers`（`ff02::1:2`）へと送信します。この Information-request メッセージには、オプション要求オプションが含まれています。Information-request メッセージを受け取ったサーバは、オプション要求オプションに対応する情報を Reply メッセージに含めて、クライアント宛のユニキャストで応答します。この応答を受け取ったクライアントは、Reply メッセージに記載された情報を自身のネットワーク接続に利用します。

ステートレス DHCPv6 の実行に最低限必要な DHCPv6 メッセージのオプションは、以下の 4 種類です。

[†16] RFC 4122：P. Leach, M. Mealling, R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace”, 2005 年 7 月



▶ 図8.16　ステートレスDHCPv6

- オプション要求オプション

  クライアントからのInformation-requestメッセージに含まれます。

- Elapsed Timeオプション

  クライアントからのInformation-requestメッセージに含まれます[†17]。

- ステータスコードオプション

  サーバからの応答であるReplyメッセージに含まれます。

- Server Identifierオプション

  クライアントに対する応答を行っているサーバに関する識別情報を伝えるために利用されます。

さらにRFC 8415では、Information-requestメッセージで要求可能な設定情報のためのオプションとして、DNS recursive name servers、DNS domain search listの2種類を紹介しています[†18]。

NOTE

8.6.2項で説明するIAアドレスオプションについては、Information-requestメッセージに含まれている場合、DHCPv6サーバでそのメッセージを破棄しなければなりません（RFC 8415のSection 16.12を参照）。

†17　RFC 3315では必須とされていませんでしたが、RFC 3315を廃止するRFC 8415では必須とされています。

†18　RFC 3736ではSIP Serversを含めた3種類が例として示されていましたが、RFC 8415ではDNS recursive name serversとDNS domain search listの2種類のみが示されています。

cc BY NC SA

8.5.1　DNS Recursive Name Serversオプション

DHCPv6でDNSサーバのIPv6アドレスを示すために利用されるDNS Recursive Name Serversオプションは、RFC 3646[†19]で定義されています。

図8.17にDNS Recursive Name Serversオプションのフォーマットを示します。DNS Recursive Name Serversオプションのoption-codeは23です。このオプションのデータ部分には、DNSサーバのIPv6アドレスを必要な数だけ含めます。したがって、オプションのデータ部分の長さを示すoption-lenは、16オクテット（128ビット）の倍数になります。

0–15	16–31
OPTION_DNS_SERVERS（23）	option-len
DNSサーバのIPv6アドレス	
DNSサーバのIPv6アドレス	
⋮	

▶ 図8.17　DNS Recursive Name Serversオプション

8.5.2　DNS Search Listオプション

DNS Search Listオプションは、ユーザがDNSを利用して名前解決を行うときに利用できるサーチリストを含めるためのオプションです。

図8.18にDNS Search Listオプションのフォーマットを示します。DNS Search Listオプションのoption-codeは24です。オプションのデータ部分には、RFC 1035[†20]のSection 3.1に示されている方法（16.3.2項を参照）でサーチリストをエンコードします。

0–15	16–31
OPTION_DOMAIN_LIST（24）	option-len
サーチリスト	
⋮	

▶ 図8.18　DNS Search Listオプション

†19　RFC 3646：R. Droms, “DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2003年12月

†20　RFC 1035：P.V. Mockapetris, “Domain names - implementation and specification”, 1987年11月

8.5.3 DNS設定のステートレスDHCPv6による方法と近隣探索メッセージによる方法との違い

RFC 8106[†21]で定義されているRouter Advertisementメッセージを利用したDNS情報の設定方法と、ステートレスDHCPv6によるDNS情報の設定方法とを比べると、ステートレスDHCPv6による方法のほうがより多くの情報を配布できるという利点があります。RFC 8106で定義されているのはDNSサーバ情報とDNSサーチリストを伝える方法だけであり、たとえばNTPサーバやSIPサーバに関する情報をRouter Advertisementメッセージで配布することはできません。DHCPv6であれば、SIPに関する情報についてはRFC 3319[†22]で定義されたオプションを利用して設定できます。したがって、DNSの設定についてはRFC 8106の方法を使う場合でも、同時にステートレスDHCPv6サーバが必要になることもありえます。もちろん、今後さらに新しいRouter Advertisementメッセージ用のオプションを定義するRFCが発行されることで、ステートレスDHCPv6との違いが少なくなる可能性も考えられます。Router AdvertisementメッセージとDHCPv6の使い分けに関しては、本書執筆時点でも議論は続いており、今後さらに状況が変わる可能性があります。

実装に依存する部分もあります。モバイル端末などで使われているAndroidは、本書執筆時点ではDHCPv6に未対応です。したがって、たとえば「ステートレスDHCPv6とRouter AdvertisementメッセージでIPv6アドレスを自動設定できるようにしよう」というネットワーク環境だと、Androidに対してキャッシュDNSサーバに関する情報を提供できません。Androidユーザに対してIPv6アドレスの自動設定環境を整えるには、RFC 6106[†23][†24]によるキャッシュDNSサーバ情報の通知が必要になります。

一方で、Router Advertisementメッセージを利用したDNS情報の設定に対応していない機器もあります。そのため、より多くの機器がIPv6環境で快適に通信できるように、DHCPv6とRouter Advertisementメッセージによる方法の両方を同時に運用することがIPv6ネットワークの運用者に求められることもあります。3種類あるIPv6アドレス自動設定手段のうちの1つだけを選んで準備すればすべてのケースに対応できるというわけではないのが現状です。ただし、たとえばmacOSがOS X 10.11でRFC 6106に対応するなど、Router Advertisementメッセージを利用した方法に対応する実装も増えているので、数年後に状況が多少変わっている可能性もあります。

8.6 ステートフルDHCPv6

DHCPv6でも、IPv4用のDHCPと同じように、DHCPサーバ側でIPv6アドレスのリース期間を制御可能な方式もあります。この方式をステートフルDHCPv6と呼びます。ステートフル

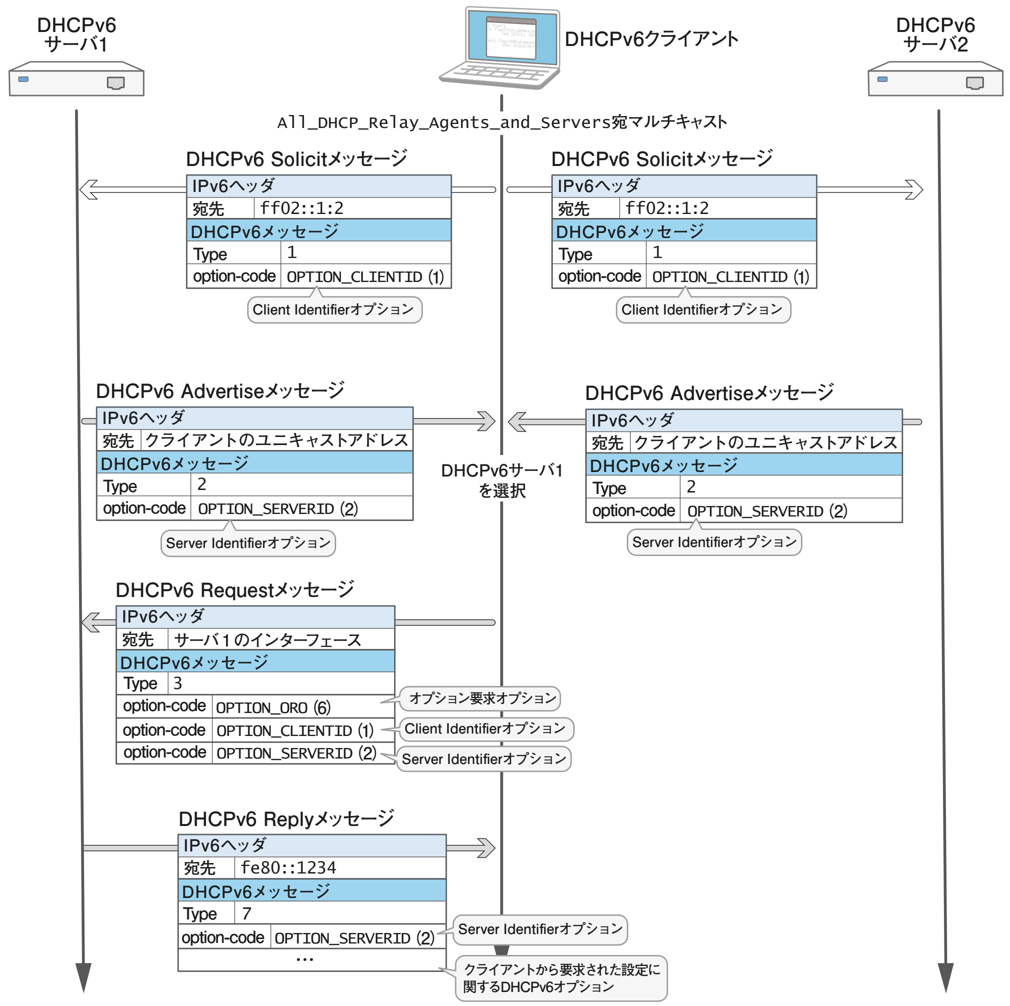
†21 RFC 8106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2017年3月

†22 RFC 3319：H. Schulzrinne, B. Volz, “Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers”, 2003年7月

†23 RFC 6106：J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, 2010年11月

†24 RFC 6106はRFC 8106で上書きされていますが、実装は旧RFC 6106をベースにしているので、あえてRFC 6106と書いています。

DHCPv6では、図8.19のように、4つのメッセージによってホストのIPv6アドレスを設定します。



▶ 図8.19 ステートフルDHCPv6

DHCPv6クライアントは、DHCPv6サーバを探すために、まず`All_DHCP_Relay_Agents_and_Servers`（`ff02::1:2`）マルチキャスト宛のSolicitメッセージを送信します。クライアントからのSolicitメッセージには、クライアントが生成したトランザクションIDと、Client Identifierオプションが含まれます。クライアントが利用を希望するIPv6アドレスがある場合には、8.6.1項で説明するIdentity Associationに関するオプションを利用して、その希望をサーバに伝えることもできます。

Solicitメッセージを受け取ったDHCPv6サーバは、Advertiseメッセージをユニキャストで DHCPv6クライアントに送信します。Advertiseメッセージには、Server Identifierオプションが含まれます。クライアントからのSolicitメッセージにIdentity Associationが含まれる場合には、サーバからのAdvertiseメッセージにもIdentity Associationを含めます。クライアントに対してそのサーバではIPv6アドレスの割り当てを行わない場合には、サーバからの

CC BY NC SA

Advertiseメッセージに含めるステータスコードオプションをNoAddrsAvailに設定します。

Advertiseメッセージを受け取ったDHCPv6クライアントは、IPv6アドレスの割り当てを依頼するDHCPv6サーバを選択し、そのサーバに対してユニキャストでRequestメッセージを送信します。このRequestメッセージには、クライアントが選択したサーバのServer Identifierオプション、クライアントのClient Identifierオプション、それにオプション要求オプションを必ず含めます。

Requestメッセージを受け取ったDHCPv6サーバは、IPv6アドレスなどの必要な情報が記載されたReplyメッセージをDHCPv6クライアント宛に送信します。このReplyメッセージには、Server Identifierオプションが必ず含まれます。また、Identity Associationやクライアントが要求した設定情報に関連するオブジェクトも含まれます。

Replyメッセージには、DHCPv6サーバによって指定されたIPv6アドレスの利用可能時間が記載されています。DHCPv6クライアントは、IPv6アドレスのリース期間を延長したい場合、DHCPv6サーバに対してRenewメッセージを送信する必要があります。

8.6.1 Identity Association

DHCPv6には、**Identity Association**という仕組みがあります。Identityは、識別という意味を持つ英単語です。Associationは、群集やつながり、関連という意味を持つ英単語です。それらを組み合わせたIdentity Associationは、クライアントに対して割り当てられたIPv6アドレスの集合として定義されています（RFC 8415）。IPv6ではインターフェースとアドレスが1対1に対応するとは限らないので、DHCPv6ではIdentity Associationという概念を利用してクライアントに割り当てるアドレスを集合として管理できるようにしているのです。

DHCPv6のクライアントでは、各ネットワークインターフェースを最低でも1つのIdentity Associationに関連づける必要があります。あるIdentity Associationを複数のネットワークインターフェースと関連づけることはできません。ネットワークインターフェースに対してIPv6アドレスなどの情報を設定する際には、Identity Associationに対してサーバから情報が割り当てられます。

各Identity Associationは、クライアント内で一意の識別子により区別されます。この識別子はIAIDと呼ばれ、クライアントが定義します。IAIDには、DHCPv6クライアントが再起動したとしても同じ値が使われ続けることが求められます。そのためRFC 8415では、毎回同じIAIDを自動的に生成する手法や、IAIDを保存する方法についても言及されています。

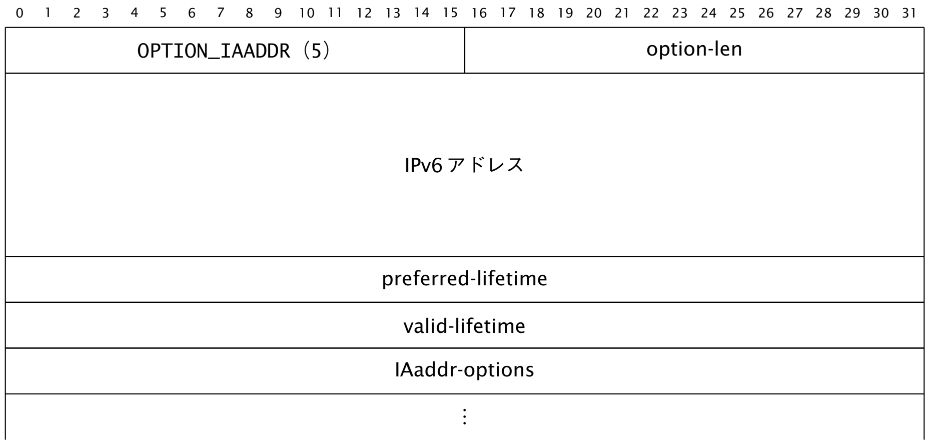
Identity Associationには、大きく分けて2つの種類があります。Temporary IPv6アドレスを扱うIA_TA（Identity Association for Temporary Address）と、一時的ではない通常のIPv6アドレスを扱うIA_NA（Identity Association for Non-temporary Address）です。これらはRFC 8415で定義されているIdentity Associationです。

IPv6アドレスではなく、IPv6プレフィックスを委任するIA_PD（Identity Association for Prefix Delegation）というIdentity Associationもあります。

8.6.2　IA アドレスオプション

IA アドレスオプションは、後述する IA_NA オプションと IA_TA オプションの中でのみ利用されるオプションです。IA_NA オプションもしくは IA_TA オプションで IPv6 アドレスを指定するのに使います。

図 8.20 に IA アドレスオプションのフォーマットを示します。IA アドレスオプションの option-code は `OPTION_IAADDR`（5）です。



▶ 図 8.20　IA アドレスオプション

IA アドレスオプションには IPv6 アドレスを含めるフィールドがあります[†25]。それに続く preferred-lifetime フィールドおよび valid-lifetime フィールドは、それぞれ当該の IPv6 アドレスの推奨期間と有効期間を表します。これらの期間の意味は、SLAAC に使われる近隣探索プロトコルのプレフィックス情報オプションにおける Preferred Lifetime および Valid Lifetime と同様です（6.5.3 項参照）。

IAaddr-options フィールドには、この IA アドレスオプションに含まれる IPv6 アドレスに関するオプションを含めます。たとえば、当該の IPv6 アドレスに関して何らかのステータスを伝える場合には、IAaddr-options フィールドにステータスコードオプションを含めます。

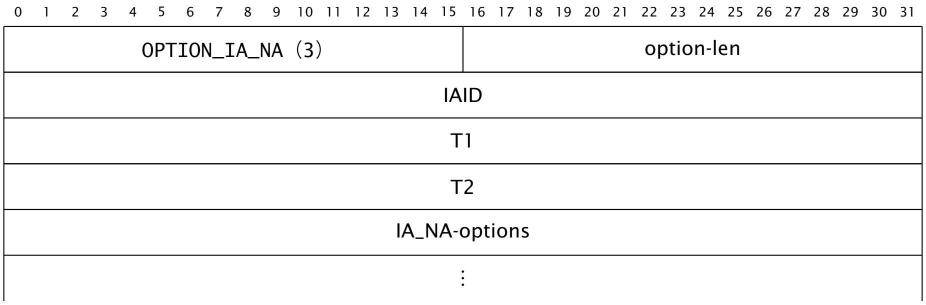
8.6.3　IA_NA オプション

IA_NA オプションは、Identity Association に対して一時的ではない IPv6 アドレスを設定するときに使うオプションです。

図 8.21 に IA_NA オプションのフォーマットを示します。IA_NA オプションの option-code は `OPTION_IA_NA`（3）です。

IA_NA には、IA アドレスオプションで指定されている有効期間（valid-lifetime）を超えてしまう前にクライアントからサーバに対して IPv6 アドレスの有効期間の延長を要求するタイミングをサーバ側で指定できる仕組みがあり、そのために使う T1 および T2 という値を秒単位

[†25] RFC 8415 では、「IPv6 address」のフィールドで示される IPv6 アドレスのプレフィックス長は /128 であるとしています。“A client MUST NOT form an implicit prefix with a length other than 128 for this address.”



▶ 図 8.21　IA_NA オプション

で設定するためのフィールドが用意されています。クライアントは、IPv6 アドレスが割り当てられてから T1 秒が経過した時点で、割り当てを行ったサーバに対して Renew メッセージを送ります。割り当てを行ったサーバが Renew メッセージに応答せずに T2 秒が経過した場合、クライアントは別のサーバに対して Rebind メッセージを送ります。これにより、最初に割り当てを受けたサーバとは別のサーバに対し、利用中の IPv6 アドレスを関連づけたうえで利用の継続を要求できます。

T1 フィールドと T2 フィールドの値が 0 の場合は、これらの値を特に設定しないことを意味します。`0xffffffff` の場合は「無限」を示し、その場合にはクライアントからの Renew メッセージと Rebind メッセージは発行されません。

T1 の値として、RFC 8415 では、Identity Association に含まれるアドレスの推奨期間（preferred-lifetime）のうちで最も短い値の 50% が推奨されています。T2 の値としては 80% が推奨されています[†26]。

IA アドレスオプションなど、IA_NA オプションのために必要なオプションは、IA_NA-options フィールドに含めます。1 つの DHCPv6 メッセージに複数の IA_NA オプションを含めることも可能です。

8.6.4　IA_TA オプション

DHCPv6 には、RFC 8981[†27] で定義されている Temporary IPv6 アドレスの割り当てをクライアントが要求する仕組みもあります。この仕組みでステートフルな DHCPv6 が利用されるときも、Temporary IPv6 アドレスと同等な仕組みを実現できるように、IA_TA オプションが存在しています。

図 8.22 に IA_TA オプションのフォーマットを示します。IA_TA オプションの option-code は `OPTION_IA_TA`（4）です。

IA_TA オプションには、IA_NA オプションと違い、T1 フィールドと T2 フィールドがありません。有効期間を超えて IPv6 アドレスを使い続ける場合、クライアントは任意のタイミング

†26　RFC 8415 の Section 22.4 より。
†27　RFC 8981：F. Gont, S. Krishnan, T. Narten, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6”, 2021 年 2 月

CC BY NC SA

0–15	16–31
OPTION_IA_TA (4)	option-len
IAID	
IA_TA-options	
⋮	

▶ 図 8.22 IA_TA オプション

で Renew を送れます。

IA アドレスオプションなど、IA_TA のために必要なオプションは、IA_TA-options フィールドに含めます。

NOTE

IAID の識別子空間は、IA_TA と IA_NA で互いに独立しています。

8.7 DHCPv6-PD

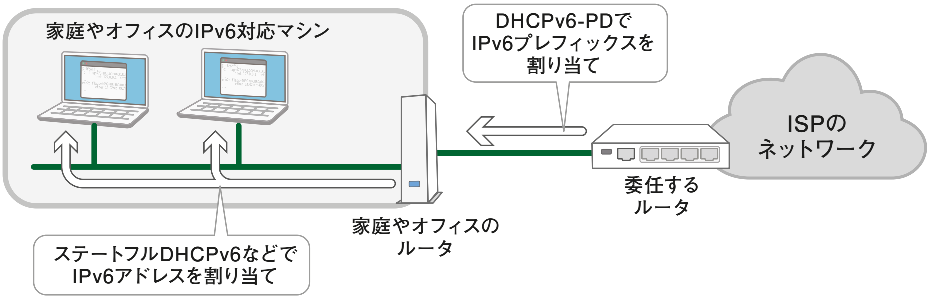
IPv6 用の DHCPv6 には、DHCPv6 サーバが DHCPv6 クライアントに対して IPv6 プレフィックスを委任（delegate）する **DHCPv6-PD**（DHCPv6-Prefix Delegation）という仕組みがあります。DHCPv6-PD には、DHCPv6 サーバと DHCPv6 クライアントの両方がルータであることが想定されているという大きな特徴があります。

DHCPv6-PD は、ブロードバンドルータなどの CPE に対して IPv6 アドレスプレフィックスを割り当てる用途に使われます。IPv4 では ISP と契約を行っているユーザに対して 1 つの IPv4 アドレスを割り当てる方式が一般的ですが、IP アドレス空間が大きい IPv6 では単一の IPv6 アドレスではなく IPv6 プレフィックスごとユーザに割り当てる運用が一般的です。

家庭やオフィスに設置されたルータに対する DHCPv6-PD の利用イメージは図 8.23 のようになります。ルータの WAN 側インターフェースでは、DHCPv6-PD クライアントとして、ISP から IPv6 プレフィックスとそのプレフィックスを利用可能な期間についての情報を受け取ります。ISP から IPv6 プレフィックスを DHCPv6-PD で受け取るときは、ステートフル DHCPv6 と同様に、クライアントからの Solicit メッセージ、サーバからの Advertise メッセージ、クライアントからの Request メッセージ、サーバからの Reply メッセージという流れになります。

NOTE

図 8.23 では、ルータが LAN 側インターフェースでステートフル DHCPv6 サーバとしてユーザに IPv6 アドレスを割り当てていますが、DHCPv6-PD を利用するときに LAN 側でステートフル DHCPv6 もしくはステートレス DHCPv6 が必須というわけではありません。LAN における IPv6 アドレス自動設定と WAN 側での DHCPv6-PD は独立であり、ISP から家庭に対するプレフィックス割り当てに DHCPv6-PD を活用しつつ、宅内 LAN 側を Router Advertisement による IPv6 アドレス自動設定で運用することも可能です。



▶ 図 8.23　DHCPv6-PD の利用例

8.7.1　IA_PD オプション

DHCPv6-PD を実現するための IA_PD オプションも、RFC 8415 で定義されています[†28]。

NOTE

IAID の識別子空間は、IA_PD、IA_TA、IA_NA で互いに独立しています。

図 8.24 に IA_PD オプションのフォーマットを示します。IA_PD オプションの option-code は OPTION_IA_PD（25）です。

0–15	16–31
OPTION_IA_PD（25）	option-len
IAID	
T1	
T2	
IA_PD-options	
⋮	

▶ 図 8.24　IA_PD オプション

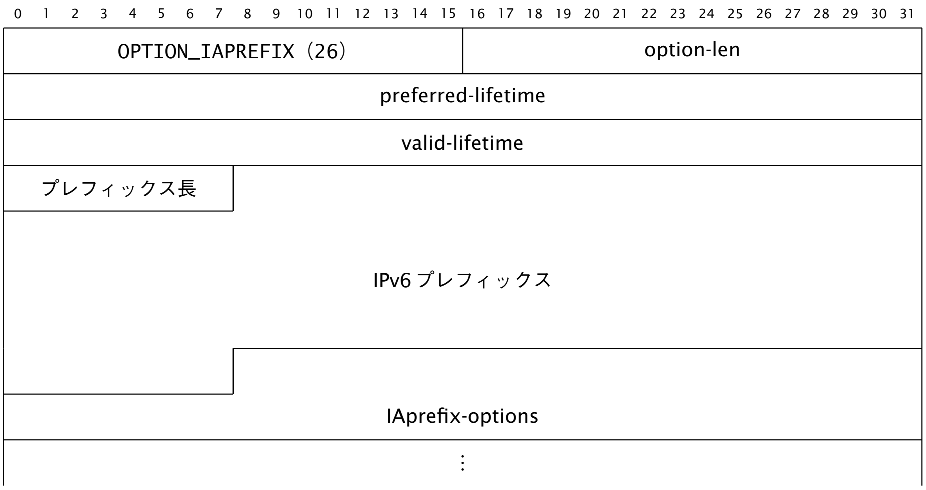
IA_PD オプションのフォーマットは IA_NA オプションに似ており、IAID、T1、T2 の用途は IA_NA オプションと同様です。IA_NA オプションとの大きな違いは、T2 フィールドに続くフィールドで運ぶオプションの中身です。IA_NA オプションには IPv6 アドレス情報を含む IA アドレスオプションが含まれますが、IA_PD オプションには、IPv6 プレフィックス情報を含む IA_PD プレフィックスオプション（後述）が含まれます。

[†28] 以前は DHCPv6 を定義している RFC 3315 と、IA_PD を定義している RFC 3633 が独立した RFC として発行されていましたが、RFC 3315 と RFC 3633 を含む DHCPv6 関連の古い RFC を廃止して更新する RFC 8415 に統合されました。

8.7.2 IA_PDプレフィックスオプション

IA_PDプレフィックスオプションは、IA_PDオプションのIA_PD-optionsフィールドで使われます。

図8.25にIA_PDプレフィックスオプションのフォーマットを示します。IA_PDプレフィックスオプションのoption-codeはOPTION_IAPREFIX（26）です。



▶ 図8.25　IA_PDプレフィックスオプション

IA_PDプレフィックスオプションの大きな特徴として、1オクテットのプレフィックス長フィールドが含まれる点が挙げられます。このオプションが運んでいるものはIPv6アドレスではなくIPv6プレフィックスです。

IAprefix-optionsフィールドには、IPv6プレフィックスに関連する設定情報を含められます。

■ DHCPv6 Prefix-Length Hintが抱える課題

IA_PDプレフィックスオプションは、IA_PDオプションに含まれる形で利用されます。しかしRFC 8415では、クライアント側がサーバに対して提供する情報として、DHCPv6-PDにおけるSolicitメッセージにIA_PDプレフィックスオプションを含めることも可能であるとしています。とはいえ、サーバ側の事情によってはクライアントから要求されたプレフィックスの割り当てが必ずしも可能とは限りません。サーバは、クライアントが提供したプレフィックス長に関するヒントを無視することもできます。

そのような不整合が発生したときの挙動は、RFC 8415には明記されていません。クライアントからのメッセージに含まれるIA_PDプレフィックスオプションが示すプレフィックス長が合わない場合の仕様としては、RFC 8168[†29]が示されています。

†29 RFC 8168：T. Li, C. Liu, Y. Cui, “DHCPv6 Prefix-Length Hint Issues”, 2017年5月

8.7.3 DHCPv6-PDのプレフィックスを除外するオプション

IA_PDプレフィックスオプションを最初に規定したRFC 3633のSection 12.1にあるDHCPv6-PDの仕様では、要求側ルータ（requesting router）に対して委任されたプレフィックスを委任元ルータ（delegating router）が利用できないという制約があります[†30]。そのため委任元ルータは、1つの顧客に対し、「委任元ルータと要求側ルータをつなぐリンク」と「要求側ルータの向こう側にある顧客サイト」という集約不能な2つの経路を持つことになります。モバイルルータなど、要求側ルータとしてDHCPv6-PDクライアントが/64プレフィックスを1つしか使わない環境では、経路を集約（aggregate）可能であるほうが望ましい場合もあります。

RFC 3633の制約を緩和して経路の集約を可能にするために、RFC 6603[†31]では、DHCPv6-PDにおいて特定のIPv6プレフィックスを除外できるPrefix Excludeオプションが規定されています。Prefix Excludeオプションにより、委任元ルータから要求側ルータに伝えられる委任プレフィックスのうち「要求側ルータの下流で使ってはならないプレフィックス」を指定できます。使ってはならないプレフィックスとして指定されるのは、「委任元と要求側の間でDHCPv6-PDのやり取りをするリンク」で使われるプレフィックスです。

Prefix Excludeオプションを受け取ったDHCPv6-PDクライアントは、下流のサブネットに除外されたIPv6プレフィックスを広告しないことが求められます。

---

†30　RFC 3633は、2018年に発行されたRFC 8415によって廃止されています。

†31　RFC 6603：J. Korhonen, T. Savolainen, S. Krishnan, O. Troan, “Prefix Exclude Option for DHCPv6-based Prefix Delegation”, 2012年5月

CC BY NC SA

第9章

IPフラグメンテーション

リンク層で一度に転送できる上位層のデータの最大長のことを**MTU**（Maximum Transmission Unit）と呼びます。たとえば、通常のイーサネットで単一のイーサネットフレームが運べるペイロードの最大長は1500オクテットなので、イーサネットのMTUは1500オクテットです。また、かつてよく利用されていたX.25パケット交換網のMTUは576オクテットです（昨今はこうした小さなMTUで運用されているリンクはほとんど存在していないと思われます）。

インターネットプロトコルでパケットを転送するとき、パケットの長さが下位のリンク層プロトコルのMTUより大きければ、パケットを分割するしかありません。このようなMTUの制限によるIPパケットの分割を**IPフラグメンテーション**と呼んでいます。IPフラグメンテーションが発生すると、元の大きなIPパケットが、複数の小さなIPパケットに分割して転送されます。この個々の小さなIPパケットのことをフラグメントと呼ぶことがあります。

IPv4では、経路途中のリンクにMTUが小さいものがあると、経路途中のルータにおいてIPフラグメンテーションが行われる場合があります。一方、IPv6では経路途中でのIPフラグメンテーションが廃止され、送信元ノードでのみIPフラグメンテーションが行われます。

本章では、まずIPv4におけるIPフラグメンテーションの基本と課題について解説してから、IPv6における仕様を見ていきます。

9.1 IPv4におけるフラグメンテーション

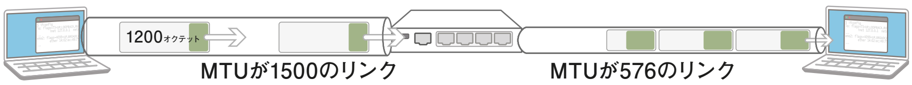
1981年に発行されたRFC 791[†1]には、以下のようにあり、IPv4が作られた当初はフラグメンテーションがインターネットの根幹を成す機能のひとつであったことがわかります。

> The internet protocol implements two basic functions: addressing and fragmentation.
>
> （参考訳）インターネットプロトコルでは2つの基本的な機能を実装する。すなわち、アドレッシングとフラグメンテーションである。

[†1] RFC 791：J. Postel, “Internet Protocol”, 1981年9月

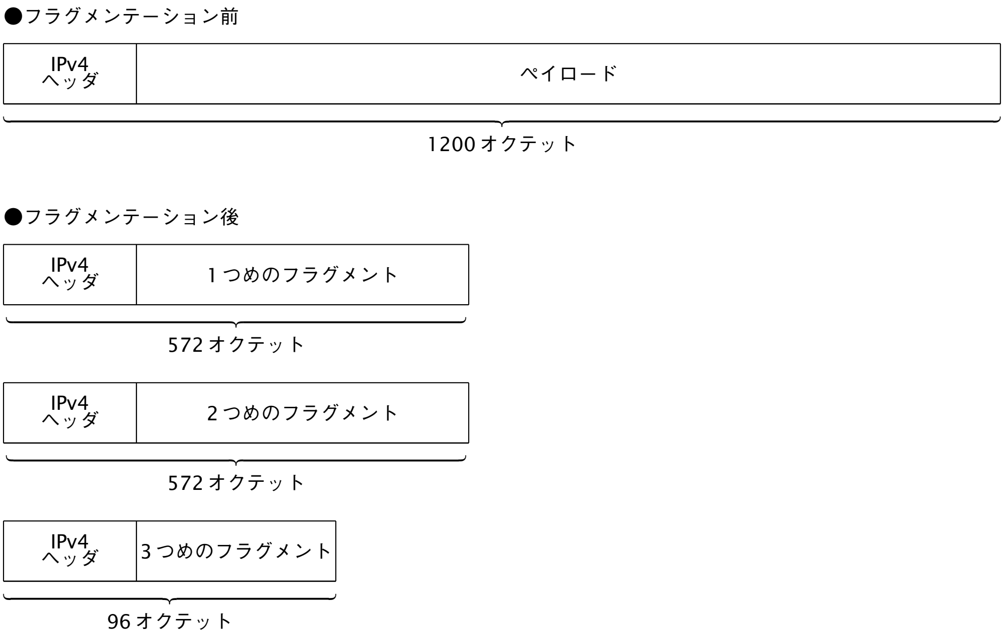
CC BY NC SA

実際、IPv4では、リンク層に合わせて経路の途中でもフラグメンテーションを行います。図9.1の例ではMTUが1500のリンクと576のリンクが混在しています。ここを1200オクテットのIPv4パケットが転送される場合、このパケットはMTUが1500のリンクではそのまま転送されますが、MTUが576のリンクではフラグメンテーションが発生する場合があります。



▶ 図9.1　IPv4でのフラグメンテーション例

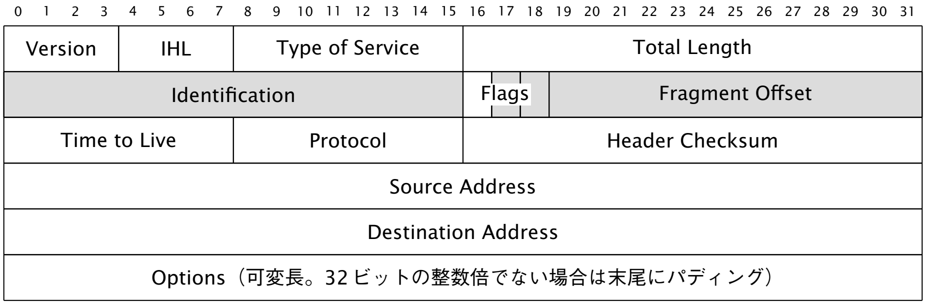
図9.1の例で、1200オクテットのIPv4パケットが3つのフラグメントに分割されると、図9.2のような状況になります（各フラグメントを表す模式図の比率は厳密ではありません）。図9.2では、MTUが576のリンクを通じてパケットを送信するために、経路途中のルータが1200オクテットのパケットを572オクテットのパケット2つと96オクテットのパケット1つに分割しています。このとき、各フラグメントには個別にIPv4ヘッダが付加されます。IPv4ヘッダ全体の長さは、この例では各IPv4ヘッダにオプションが付いていないので、20オクテットです。IPv4ではフラグメントの境界は8オクテット単位とされているので（RFC 791のSection 3.2）、この場合のIPヘッダを含めたフラグメンテーション後のパケットの大きさはMTUと同値の576オクテットではなく、20 + 552 = 572オクテットとなります。



▶ 図9.2　IPv4のフラグメンテーションのようす

9.1.1 IPv4のフラグメンテーションで利用されるIPv4ヘッダのフィールド

IPv4は経路途中でのフラグメンテーションを前提としたプロトコルなので、IPv4ヘッダにそのためのフィールドが存在します。具体的には、Identificationフィールド、Flagsフィールド、Fragment Offsetフィールドの3つです（図9.3）。



▶ 図9.3 IPv4でのフラグメンテーションで利用されるIPv4ヘッダフィールド

IPv4ヘッダのIdentificationフィールドには、IPパケットがフラグメントに分割されるときに、各フラグメントが同じIPパケットであったことを識別するための値が設定されます。各フラグメントのIdentificationフィールドが同じ値になるので、それらを受け取った側で分割されたフラグメントを再構築するのに利用できます。なお、IPv4のIdentificationフィールドの仕様はRFC 6864[†2]によって更新されています。

Fragment Offsetフィールドは、各フラグメントが元のIPパケットのどの部分かを示すためのものです。IPv4では8オクテット単位でパケットをフラグメンテーションすることになっているので、このフィールドの値も8オクテット（64ビット）単位で指定します。先頭のフラグメントについてはFragment Offsetフィールドをゼロに設定します。

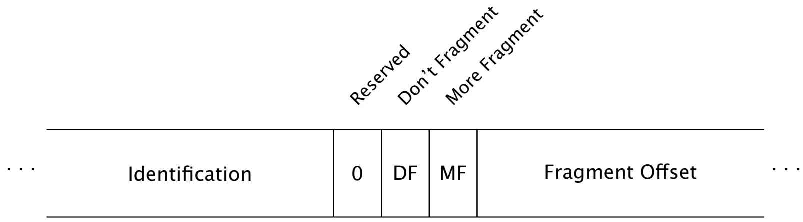
Flagsフィールドは図9.4のように区分されており、2ビットめがフラグメンテーションを許可するかどうかを示すフラグです。このフラグはDon't Fragmentビット（DFビット）と呼ばれ、このビットが1となっているパケットについてはフラグメンテーションが禁止されます。

Flagsフィールドの3ビットめは、最後のフラグメントでないかどうかを示すフラグで、More Fragmentsビット（MFビット）と呼ばれます。このビットが0のパケットは、一連のフラグメントのうち最後のパケットであるか、そもそもフラグメンテーションが行われていないパケットであることを示します。このビットが1のパケットは、フラグメンテーションされたパケットの一部であり、さらにフラグメントが続くことを示します。

9.1.2 IPフラグメンテーションとTCP MSS

本書執筆時点におけるIPv4インターネットを流れるパケットの大部分がTCPによるものであると言われています。TCPでは、データを**セグメント**と呼ばれる小さな単位に分割して送信します。そして、セグメントの最大サイズ（**MSS**、Maximum Segment Size）を設定する仕組

[†2] RFC 6864：J. Touch, “Updated Specification of the IPv4 ID Field”, 2013年2月



▶ 図9.4　IPv4ヘッダのフラグフィールド

みがあります（RFC 879[†3]およびRFC 1191[†4]）。

MSSは、TCPオプションとして、TCPコネクション確立時に「自分が受信可能なセグメントサイズ」として送信されます。オプションなので利用が必須ではありませんが、現在はほとんどの実装でMSSが利用されています。その情報をもとに、送信データを事前分割してからTCPでパケットを送信します。

TCPが送信するMSSオプションに含まれる情報は、基本的にその機器に設定されたMTUから算出されるので、途中経路上にMSSよりもMTUが小さい箇所があればフラグメンテーションが発生する可能性があります。しかし現在のインターネットは、サーバが接続されている部分、インターネットのコア部分、家庭内ネットワークの部分など、その大半が1500オクテットのMTUで運用されています。つまり、TCPを利用したときにフラグメンテーションが発生しがちな箇所は、ある程度限定されています。

ユーザに気づかれずにフラグメンテーションが頻繁に発生する状況としては、PPPoEやIPトンネルなどが運用されることによってMTUが1500オクテットよりも多少小さくなる環境で、かつ、ユーザがNATルータの裏側でインターネットを利用している場合が考えられます。図9.5のように、NATルータがPPPoEによってインターネットに接続している場合を考えてみましょう。この場合、PPPoEヘッダの6オクテットとPPPヘッダの2オクテットの合計8オクテットが1500オクテットの先頭にきます。そのため、その後のペイロードは1492オクテットとなります。ここに1500オクテットのパケットが到着すると、ISP側のルータがフラグメンテーションを行います。ISPに接続されたNATルータは、2つにフラグメント化されたパケットを受け取りますが、インターネットから見るとNATルータはエンドノードでもあるので、2つのフラグメントを再構成して1500オクテットのパケットにしたうえで、IPアドレスなどのフィールドを変更してユーザへと転送します。

このような環境では、ユーザは何事もなかったかのように1500オクテットのパケットを受け取るので、途中経路上でフラグメンテーションが発生していることにまったく気づかないままインターネットを利用できてしまいます。

9.2　IPv6フラグメントヘッダ

IPv6でのフラグメンテーションは、図9.6のようなIPv6拡張ヘッダのひとつである**フラグメントヘッダ**を利用して行われます。IPv6拡張ヘッダは、経路の途中にあるノードでの処理、挿

†3　RFC 879：J. Postel, “The TCP Maximum Segment Size and Related Topics”, 1983年11月
†4　RFC 1191：J.C. Mogul, S.E. Deering, “Path MTU discovery”, 1990年11月



▶ 図 9.5 IPv4 NAT によるフラグメントの再構築

入、削除が禁止されています[†5]。フラグメントヘッダも例外ではありません。そのため、IPv6 でのフラグメンテーションはエンドノード同士で行うことになります。この点が、経路途中のルータでフラグメンテーションが発生する IPv4 との大きな違いです。

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Next Header	予約	Fragment Offset	予約	M
Identification				

▶ 図 9.6 フラグメントヘッダ

- Next Header（8 ビット）

  この拡張ヘッダの直後にくるプロトコルの種類を示します。IPv6 ヘッダにおける Next Header フィールドと同じく、IANA によって割り当てられている値（IPv4 の Protocol Type フィールドで利用される値と同じ値）を指定します。

- 予約（8 ビット）

  将来の利用に備えたフィールドです。

- Fragment Offset（13 ビット）

  後述するように、このフラグメントヘッダの後には、フラグメント化されたデータが続きます。そのデータが、フラグメント化される前のデータの先頭から見てどの部分のデータなのかを、8 バイト単位で示すフィールドです。

- 予約（2 ビット）

  将来の利用に備えたフィールドです。

- M フラグ（1 ビット）

  このフラグメントが、最後のフラグメントか、それとも後続するフラグメントがあるかを示すビットです。最後のフラグメントなら 0、後続するフラグメントがあるなら 1 です。

---

†5 ただし、Hop-by-Hop オプションに関しては途中経路上での確認や処理が可能です。

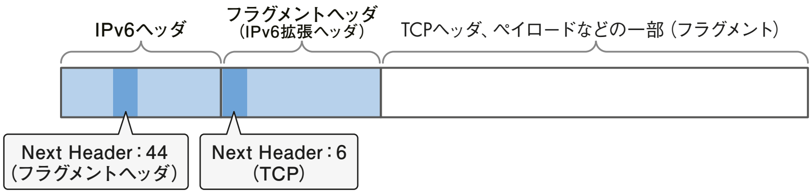
CC BY NC SA

- Identification（32ビット）

  送信元ノードは、フラグメンテーションが行われるパケットごとに、フラグメント識別子の値を生成してこのフィールドに格納します。宛先ノードでは、宛先アドレスと送信元アドレスが同じで、フラグメント識別子の値も同じフラグメントを、1つのパケットとして再構築します。送信元ノードでは、同じ宛先への送信に「つい最近」使ったフラグメント識別子の値とは異なる値を選択する必要があります。「つい最近」の定義として、RFC 8200では、「送信元から宛先までの転送時間および同一のパケットの他のフラグメントが再構築されるまでの時間を含む、パケットが生存するであろう時間の最大値の範囲内」としています。

9.2.1 IPv6フラグメンテーションの実際

例として、1つのTCPパケットが複数のIPv6パケットへとフラグメント化されて送信されるケースを考えてみましょう。図9.7に、フラグメント化されたTCPパケットが運ばれるときのIPv6パケットのようすを示します。



▶ 図9.7 IPv6パケットにおけるフラグメントヘッダの使い方

図9.7を見るとわかるように、IPv6ヘッダの直後には、拡張ヘッダとしてフラグメントヘッダが続きます。

フラグメントヘッダのプロトコル番号は44なので（107ページの表4.1参照）、フラグメントヘッダの前のヘッダのNext Headerフィールドには、フラグメントヘッダを示す44という数値が入ります。したがって、図9.7の例でも、IPv6ヘッダのNext Headerフィールドには44という数値が入っています。

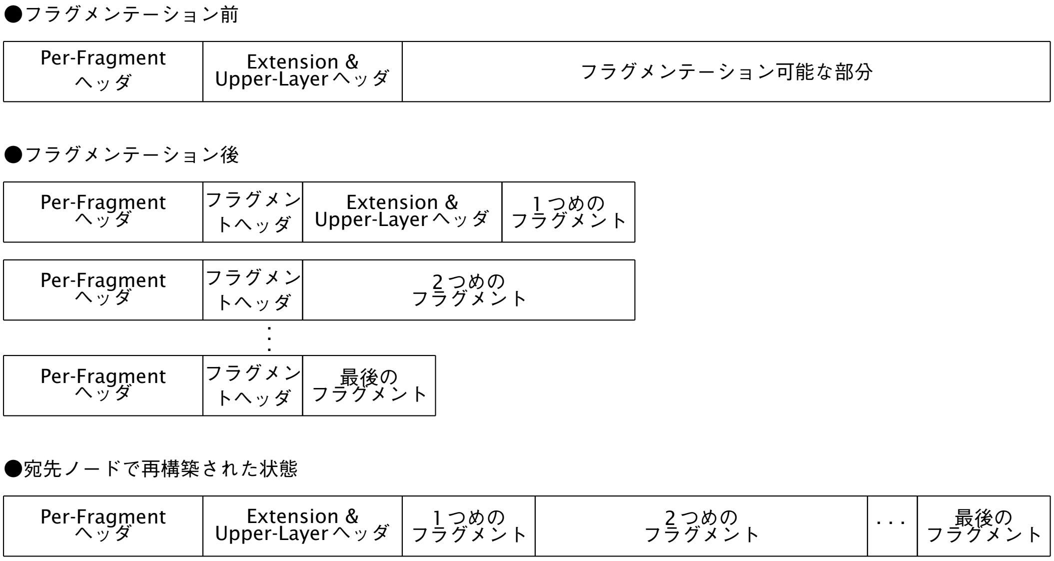
図9.7は、フラグメント化された個々のデータがIPv6パケットとしてどのように運ばれるかを示したものです。次は、元の大きなIPv6パケットが送信元のノードでどのようにフラグメント化され、宛先のノードでどのように再構成されるかを見てみましょう。図9.8に、フラグメント化される前のパケットと、分割されたフラグメントのようすを示します。さらに、それらのフラグメントが再構成されるようすも示します。

図9.8における「Per-Fragmentヘッダ」および「Ext & Upper-Layerヘッダ」の内容を下記にまとめます。

- Per-Fragmentヘッダ

  元のIPv6パケットには、IPv6ヘッダが付いています。この部分はフラグメント化せず、そのままフラグメント化した各パケットにすべて含める必要があります。

cc BY NC SA



▶ 図9.8　IPv6のフラグメンテーションのようす

また、元のIPv6パケットには、IPv6ヘッダの直後にHop-by-Hopオプションヘッダが続いている場合もあります。このIPv6拡張ヘッダの情報は、元のIPv6パケットの宛先までに存在する各ノードやルータにおける処理に必要となる情報です。そのため、Hop-by-Hopオプションヘッダがあれば、それもフラグメント化された各パケットに含めてあげる必要があります。

さらに、Hop-by-Hopオプションヘッダの直後には、宛先オプションヘッダとルーティングヘッダが続いている場合もあります。これらは、ルーティングヘッダのオプションで指定された途中経路のルータで必要になる情報なので、存在する場合にはやはりフラグメント化された各パケットにも含めてあげる必要があります。

これら、フラグメント化された各パケットに含められる元のIPv6パケットの情報（IPv6ヘッダと、もしあれば一部のIPv6拡張ヘッダ）のことを、RFC 8200[†6]では**Per-Fragmentヘッダ**と呼んでいます。

- Extension & Upper-Layerヘッダ

  IPv6パケットをフラグメント化して転送する際、経路途中のファイアウォールで上位層のヘッダ情報が必要になる場合があります。その場合、上位層のヘッダ情報がフラグメント化されて複数のパケットに分散していると、フラグメント化前の元のIPv6パケットに対して適用されるべき経路途中のファイアウォールのポリシーが適切に実行されません。そのため、2014年に発行されたRFC 7112[†7]では、上位層のヘッダまでを図9.8に示すように先頭のフラグメントにすべて付加しなければならない（MUST）としています。

---

†6　RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017年7月
†7　RFC 7112：F. Gont, V. Manral, R. Bonica, “Implications of Oversized IPv6 Header Chains”, 2014年1月

この先頭のフラグメントに含めなければならない部分には、Per-Fragmentヘッダには含まれないIPv6拡張ヘッダ（ルーティングヘッダより後ろの部分）も含まれます。この部分のことを、RFC 8200では**Extension & Upper-Layerヘッダ**と呼んでいます（RFC 8200でも、フラグメンテーションの仕様はRFC 7112を満たすものになっています）。

なお、Upper-Layerとありますが、これは必ずしもトランスポート層のヘッダを指すとは限りません。ICMPv6ヘッダの場合もあれば、他のIPヘッダの場合（IPv4やIPv6のトンネリングを行っている場合）もあります。また、Encapsulating Security Payloadヘッダがある場合は、以降のペイロードがすべて暗号化されているので、ESPヘッダまでがExtension & Upper-Layerヘッダとなります。

Extension & Upper-Layerヘッダは、先頭のフラグメントでフラグメントヘッダの後ろに配置されます。

NOTE

Per-Fragmentヘッダに相当する部分は、RFC 2460[†8]では「Unfragmentable Part（フラグメント化不能部分）」と表現されていました。また、RFC 2460にはExtension & Upper-Layerヘッダの区別もありません。
IETFのv6opsワーキンググループでは、draft-ietf-6man-rfc2460bisという形でRFC 2460を置き換えるRFCを作成するための議論が交わされていましたが、その初期バージョン（00）で「Unfragmentable Headers」および「Extension & Upper-Layer」が定義されました。さらにバージョン01で前者が「Per-Fragment Headers」という表現に変更されて、そのままRFC 8200になりました。

9.2.2　IPv6ヘッダとIPv6拡張ヘッダの最大長

IPv4ヘッダのフィールドには、可変長の「IPv4ヘッダオプション」が含まれているので、IPv4ヘッダ自体の長さも可変長です。そのためIPv4ヘッダには、IHL（Internet Header Length）という、ヘッダ長を4オクテット単位で示すフィールドが用意されています。IHLは4ビットのフィールドなので、IPv4ヘッダオプションを含めたIPv4ヘッダの最大サイズは、$15 \times 4 = 60$オクテットということになります。

一方、IPv6ヘッダは、それ自体は20オクテットの固定長です。しかし、拡張ヘッダを好きなだけ付加できます。IPv6ヘッダとIPv6拡張ヘッダを合計した長さの最大値は、RFC 2460では定義されていません。フラグメント化不能部分（IPv6ヘッダおよび経路途中のノードでの転送に必要な情報）だけを定めたRFC 2460のままでは、フラグメント化不能なIPv6拡張ヘッダが異常に長かったり各フラグメントの長さが短かったりした場合に、上位層のヘッダ情報が先頭のフラグメントに収まりきらず、結果として元のIPv6パケットのままであれば通過できなかったはずのファイアウォールを通過できてしまうといった問題が起きる可能性があります。

そこで、「IPv6パケットを送出するホストでフラグメント化する際には、上位層のヘッダまでの部分を先頭のフラグメントに収めなければならない」としてRFC 2460を更新したのが、

†8　RFC 2460：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 1998年12月

CC BY NC SA

RFC 7112です[†9]。

この制約から、IPv6ヘッダとIPv6拡張ヘッダの合計の長さも自然と制約を受けることになります。つまり、IPv6ヘッダ長との合計がPath MTU（第10章参照）を超えるようなサイズになるIPv6拡張ヘッダを付加することはできません。IPv6では、IPv4のように明示的な値ではないものの、フラグメンテーションの仕様によってIPv6拡張ヘッダに対する最大値が制約されているというわけです。

9.2.3 複数のフラグメントに重複する箇所がある場合

RFC 2460は複数のフラグメントが重複する箇所を含むことを許容する仕様になっています。2009年に発行されたRFC 5722[†10]は、この仕様を上書きします。

RFC 5722では、RFC 2460が許容している重複する箇所があるフラグメントが発生させるセキュリティ上の問題を解説しています。TCPヘッダを含む最初のフラグメントにおいて、TCPのSYNとACKのフラグを1とすることで、ファイアウォールに対しては内側からの接続要求に対する応答であると認識させます。以後、そのファイアウォールは同じ識別子を持つフラグメントをもともとは単一のパケットであったと認識する可能性があります。そのような状況で、攻撃者はTCPのSYNを1としつつ、ACKを0としたTCPヘッダを含む新たなフラグメントを送りつけることで、受信者側が持つTCPヘッダの情報を書き換えることによって、本来であれば通過させることができなかったTCPの接続要求を外部から受け付けてしまうようになります。さらに、TCPヘッダの任意のフィールドを変更することが可能であるため、TCPヘッダのフラグだけではなく、ポート番号なども変更可能です。

そのような攻撃を防ぐために、RFC 5722は、フラグメントを生成する送信側では重複する箇所を含むフラグメントを生成することを禁止し、重複する箇所を含むパケットを受け取った受信側は同じ識別子を持つ一連のフラグメントを破棄するように仕様を変更しています。RFC 8200でも、RFC 5722に記載された仕様と同じくフラグメントを破棄することを求めると同時に、そのときにはICMPv6エラーメッセージを送信しないとも記載しています。

9.2.4 Atomicなフラグメンテーション

RFC 2460よりも後に発行された次のRFCでは、「Atomic なIPv6フラグメンテーション」が扱われています。Atomic なIPv6フラグメンテーションとは、IPv6フラグメントヘッダを含むものの、パケットが分割されずに最初のフラグメントにすべてが含まれているような状態を意味します。

- RFC 6946[†11]: Processing of IPv6 “Atomic” Fragments
- RFC 8021[†12]: Generation of IPv6 Atomic Fragments Considered Harmful

---

†9 RFC 7112では、上位層のヘッダまでの部分のことを「IPv6 Header Chain」と呼んでいます（RFC 8200における Per-Fragmentヘッダおよび Extension & Upper-Layerヘッダに相当する部分）。

†10 RFC 5722：S. Krishnan, “Handling of Overlapping IPv6 Fragments”, 2009年12月

†11 RFC 6946：F. Gont, “Processing of IPv6 "Atomic" Fragments”, 2013年5月

†12 RFC 8021：F. Gont, W. Liu, T. Anderson, “Generation of IPv6 Atomic Fragments Considered Harmful”, 2017年1月

CC BY NC SA

RFC 6946はStandards Trackです。RFC 6946では、ICMPv6 Packet Too Bigメッセージで次ホップのMTUが1280オクテット未満であるという報告を受けたホストが、実際には単一の独立した（Atomicな）パケットにフラグメントヘッダを付加して後続のパケットを送信できる仕様になっていることが紹介されています（1280オクテット未満のMTUを報告するICMPv6 Packet Too Bigメッセージが発生する例としては、IPv6をIPv4に変換するような処理が挙げられています）。そのようなAtomicなフラグメントを受け取ったホストが、そのAtomicなフラグメントと「IPv6送信元アドレス、IPv6宛先アドレス、フラグメントヘッダ識別子」が一致するフラグメントをすでに受け取っていれば、フラグメントが重複していると判定され、「IPv6送信元アドレス、IPv6宛先アドレス、フラグメント識別子」が一致するフラグメントはすべて破棄されます。これは、前項で説明したRFC 5722の仕様により、重複するフラグメントは破棄されることになっているからです。RFC 6946は、そのようなことが起きないように、Atomicなフラグメントは個別のパケットとして認識することを求める仕様としてRFC 2460を上書きするものです。

RFC 8021はInformationalなRFCです。RFC 8021では、1280オクテット以下のMTUを報告するICMPv6 Packet Too Bigメッセージを偽造することによるDoS攻撃の可能性を紹介しています。RFC 7872[†13]の調査では、IPv6拡張ヘッダが付いたパケットがインターネット上で破棄される可能性について紹介していますが、フラグメントヘッダもIPv6拡張ヘッダであるため、ICMPv6 Packet Too Bigメッセージを1280オクテット未満の値で送りつけることによってフラグメントヘッダを付加させることができれば、そのトラフィックがインターネット上で破棄されることを狙うようなDoS攻撃が可能であると考えられます（4.3.7項も参照）。そのような問題点があるため、RFC 8021では、IPv4/IPv6トランスレータが1280オクテット以下のMTUを報告するICMP Packet Too Bigメッセージを送信しないようにすることを提案しています。そのうえで、その提案が実装されれば、MTUが1280オクテット未満のICMPv6 Packet Too Bigを受け取った場合の仕様を作る必要性がなくなるので、そのようなICMPv6メッセージを無視すべきであるとしています。

RFC 2460を上書き廃止するRFC 8200では、RFC 6946とRFC 8021の流れを受けて、RFC 2460にあった仕様を以下のように変更しています。

- Fragment Offsetが0かつMフラグが0であるフラグメントヘッダを受け取った場合、他のフラグメントヘッダとIPv6送信元アドレス、IPv6宛先アドレス、フラグメント識別子のセットが同じパケットを受け取ったとしても、別々に処理する仕様にしています。
- ICMP Packet Too Bigが1280未満のMTUを受け取ったときにフラグメントヘッダを付けて送信するという仕様を削除しています。

[†13] RFC 7872：F. Gont, J. Linkova, T. Chown, W. Liu, “Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World”, 2016年6月

CC BY NC SA

第10章

Path MTU discovery

1回で送信できるネットワーク層のデータの最大値のことを**MTU**（Maximum Transmission Unit）と呼びます。MTUは、通信を行う際のパフォーマンスに影響するのはもちろん、場合によっては通信障害の要因ともなります。インターネットでは複数のルータを越えてIPパケットが転送され、各ルータ間のMTUはそれぞれ異なることもあります。それらのMTUのうち最小の値より大きなサイズのIPパケットを送信すれば、そのMTUを持つリンクにおいてそのパケットは転送できません。そのような状況が発生した場合、IPv4では途中のルータでフラグメンテーションを行う可能性がありました。一方、IPv6では経路の途中にあるルータではフラグメンテーションを行いません。そのため、IPv6パケットを宛先に届けるには、通信経路における最小のMTUを事前に把握し、そのサイズを超えないように送信元でフラグメンテーションを行う必要があります。この、通信経路の最小MTUを検知する手法が、本章で解説する**Path MTU Discovery**です。Path MTU Discoveryでは、いくつかのプロトコルを組み合わせてIPパケット長の最適な値を推測します。

IPv4では、IPv4ヘッダのDF（Don't Fragment）ビットがセットされていなければ、途中経路上のルータが必要に応じてパケットのフラグメンテーションを行います。一方、IPv6では途中経路上のルータではフラグメンテーションを行わないので、適切なMTUが設定されていないパケットは通信相手に届きません。そうした問題が発生しやすい背景はありますが、Path MTU DiscoveryそのものはIPv6のための仕組みではなく、IPv4時点から存在している仕組みです。

IPv4におけるPath MTU Discoveryを記述したRFC 1191[†1]は、1990年に発行されています。IPv6については、1996年にRFC 1981[†2]が発行されていますが、このRFC 1981は2017年にRFC 8201[†3]によって上書きされています。

NOTE

Path MTU DiscoveryはIPv4でもIPv6でも必須ではありません。利用するかどうか、どのような手法で行うかは、実装もしくは設定に依存します。

†1 RFC 1191：J.C. Mogul, S.E. Deering, “Path MTU discovery”, 1990年11月
†2 RFC 1981：J. McCann, S. Deering, J. Mogul, “Path MTU Discovery for IP version 6”, 1996年8月
†3 RFC 8201：J. McCann, S. Deering, J. Mogul, R. Hinden, “Path MTU Discovery for IP version 6”, 2017年7月

10.1　Path MTU discoveryに関係するIPv4の機能

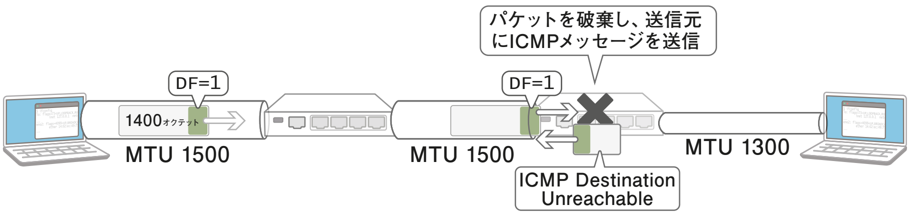
IPv4では、途中経路上のルータが必要に応じてパケットのフラグメンテーションを行います。ただし、IPv4ヘッダのフラグフィールドに含まれるDFビット（図10.1で灰色になっているビット）がセットされているパケットはフラグメンテーションが禁止されます。

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options（可変長。32ビットの整数倍でない場合は末尾にパディング）				

▶ 図10.1　IPv4ヘッダフィールド中のDFビット

IPv4では、フラグメンテーションが必要にもかかわらずDFビットがセットされているパケットをルータが受け取った場合、パケットを破棄すると同時にICMPメッセージをパケットの送信元アドレスに対して送信します（図10.2）。



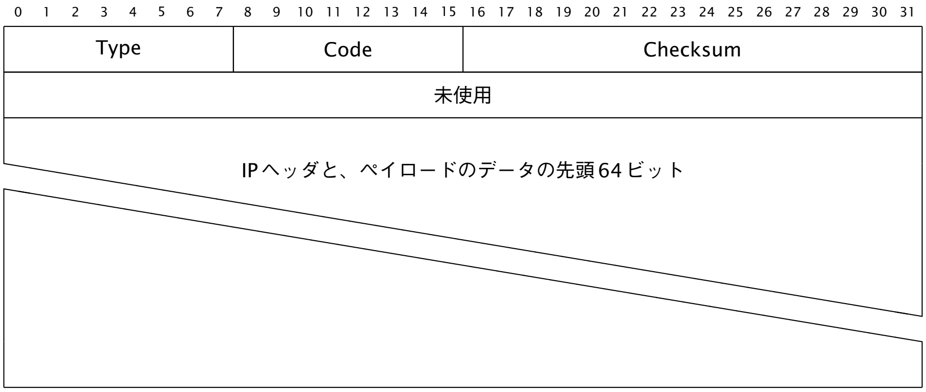
▶ 図10.2　IPv4でのPath MTU Discovery

この場合のICMPメッセージのタイプは、宛先へ到達できないことを示すDestination Unreachable（3）です。また、タイプにおける意味を示すコードは、フラグメンテーションが必要であるにもかかわらずDFビットがセットされていることを示す「fragmentation needed and DF set（4）」です。

このときのICMPメッセージのフォーマットを図10.3に示します。このICMPメッセージはIPv4ヘッダの直後に続きます。

このICMPメッセージを受け取った送信ホストは、最初に送信したDFビット付きのIPv4パケットが、通信経路の途中にあるリンクのMTUより大きかったことを知ります。しかし、IPv4のICMPにはDFビット付きパケットが破棄される原因となったリンクのMTUを通知する機能はありません。そのため、送信元ホストでは、DFビットを設定したパケットを複数送信しながらICMPメッセージの応答を確認することで、最適なPath MTUを推測する必要があります。その推測手法は明確に定義されておらず、実装依存となっています。また、Path MTUが判明

[CC BY NC SA]



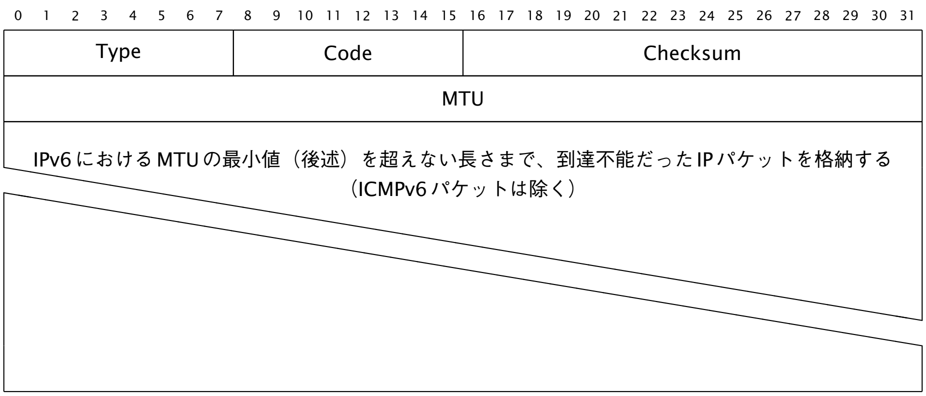
▶ 図 10.3　IPv4 ICMP Destination Unreachable メッセージフォーマット

して小さくした送信単位を再度大きくすることを試すタイミングなども実装依存になっています。

10.2　Path MTU Discovery に関係する IPv6 の機能

IPv6 ヘッダには IPv4 ヘッダの DF ビットに相当する機能はありません。ICMPv6 については、Packet Too Big メッセージ（図 10.4）が Path MTU Discovery で利用されます。ICMPv6 の Packet Too Big メッセージには、MTU の値を示すフィールドが存在しない IPv4 の ICMP メッセージとは異なり、原因となった MTU の値を示すフィールド存在します。そのため、IPv4 のように複数のパケットを送信しながら最適な Path MTU を探る必要はありません。

さらに、IPv6 の Path MTU Discovery では、IPv6 ヘッダに記載されたフローラベルごとに異なる Path MTU を利用することも可能になっています。



▶ 図 10.4　ICMPv6 Packet Too Big メッセージフォーマット

10.3　上位層プロトコルとPath MTU Discovery

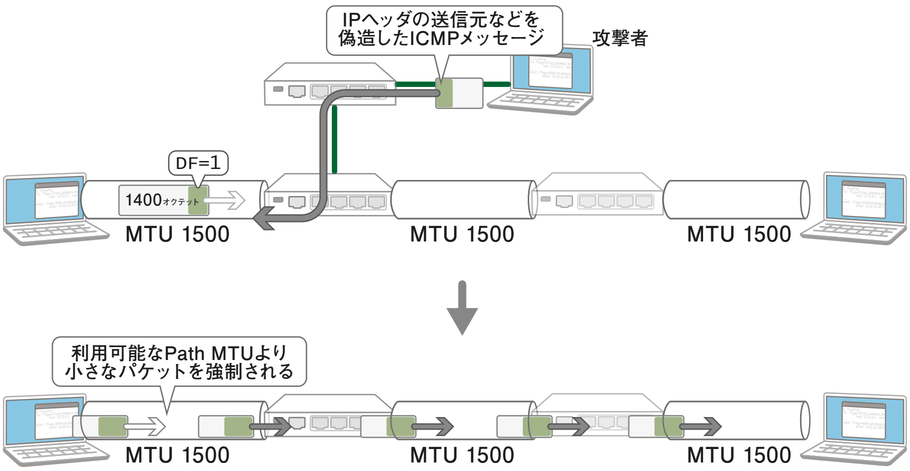
RFC 4821[†4]では、上位層におけるPath MTU Discoveryの活用方法が標準化されています。具体的には、TCP、SCTP、その他アプリケーションにおけるPath MTU Discoveryの活用手法が解説されています。

また、RFC 8201では、TCP MSSがPath MTU Discoveryによって得られる結果をもとに変更されることがあったとしても、TCP MSSとPath MTU Discoveryが互いに独立する仕組みであると記載されています。

ICMPを悪用したTCPへの攻撃手法とそれらへの対処についてまとめたRFC 5927[†5]の中でも、Path MTU Discoveryについて触れられています。RFC 5927では、TCPでのPath MTU Discoveryの活用について、ICMPによってパケット長が大きすぎることが報告されたとしても即座にTCPのMSSを変更せず、実際に大きなTCPセグメントの送信が失敗することを確認してからMSSを変更する手法が説明されています。

10.4　Path MTU Discoveryの仕組みを悪用したDoS攻撃

Path MTU Discoveryで利用されるIPv4のICMPとIPv6のICMPv6にはパケットを認証する機能がないので、第三者によるICMPメッセージの偽造が可能です。そのため、Path MTU Discoveryの仕組みを悪用したDoS攻撃が可能です。



▶図10.5　偽造されたICMPによるPath MTUの誤認

想定される攻撃は、本来の値よりも小さい値をPath MTUとして誤認させ、通信性能を低下させるというものです。たとえば、本来であれば1500オクテットであるはずのPath MTUを、IPv4であれば576オクテット、IPv6であれば1280オクテットであると誤認させることによって通信に必要となるパケット数を増やし、通信性能を低下させることができます。

†4　RFC 4821：M. Mathis, J. Heffner, “Packetization Layer Path MTU Discovery”, 2007年3月
†5　RFC 5927：F. Gont, “ICMP Attacks against TCP”, 2010年7月

(cc) BY NC SA

Path MTUを実際よりも低い値に設定されてしまうDoS攻撃は、単に通信速度を低下させるだけであり、通信ができなくなる状態を作るわけではありません。ただし、この仕組みを悪用して他の攻撃手法に組み合わせることは可能です。たとえば、UDPで送信されるDNSの応答パケットを意図的にフラグメンテーションさせたうえで2つめ以降のパケットを偽造して送りつけることでキャッシュポイズニングを行うという攻撃においてPath MTUの誤認を利用する手法が、2013年に公表されています[†6]。

逆に、Path MTUを本来よりも大きな値に誤認させることができれば、攻撃を受けた側の通信が遮断されてしまう可能性はあります。とはいえ、RFC 1191とRFC 8201には「ICMPメッセージをもとにしてPath MTUの値を上昇させてはならない」という規定があるので、そのような攻撃は成立しません。

ICMPメッセージ偽造によるPath MTU Discoveryの仕組みに対するDoS攻撃への対処は、ネットワーク層全体の仕組みとして提供されるわけではなく、Path MTU Discoveryを活用する機能ごとに行われます。

†6 Amir Herzberg, Haya Shulman, “IP fragmentation attack on DNS”, RIPE 67, 2013年10月, https://ripe67.ripe.net/presentations/240-ipfragattack.pdf

CC BY NC SA

[書籍版では空白のページです]

CC BY NC SA

第11章

IPv6マルチキャスト

マルチキャストは、「誰かがデータを送り、そのデータが必要な人にだけ届けられる」という通信方法です。マルチキャストでは、「送信者はネットワークのどこにいてもよく、受信者は複数存在する」という状況が想定されています。IPパケットのマルチキャストについては、RFC 1112[†1]でサービスモデルが定義されています。

RFC 1112によれば、IPマルチキャストでは0もしくは1以上のノードによって構成される「グループ」に対してIPパケットを送信します。このときの宛先アドレスとしては、グループを指す1つのIPアドレスを使います。このグループのことを**マルチキャストグループ**といい、マルチキャストグループを指すIPアドレスを**マルチキャストアドレス**といいます。

マルチキャストの受信側は、マルチキャストグループに「参加（join）」することで、そのグループに割り当てられたマルチキャストアドレス宛のデータを受け取れるようになります。そのマルチキャストアドレス宛のデータを受け取る必要がなくなったら、該当するマルチキャストグループから「離脱（leave）」します。ノードによるマルチキャストグループへの参加や離脱は、任意のタイミングで動的に可能です。

一方、マルチキャストの送信側は、あるマルチキャストグループを指すマルチキャストアドレスをIPパケットの宛先に設定することで、マルチキャストグループに対してデータを送信します。受信側とは違い、送信側は自分が参加していないマルチキャストグループに対してもIPマルチキャストでデータを送信することが可能です。

マルチキャストアドレスは、ユニキャストで使われるIPアドレスと異なり、特定の宛先を示すものではありません。とはいえ、マルチキャストグループを構成する各ノードに対しては、ユニキャストの場合と同様に「best-effort（最善努力）」でIPパケットが送信されます。best-effortなので、グループを構成するすべてのノードに必ずパケットが届くことが保証されているわけではありません。また、受信の順番が送信の順番と必ず一致するわけでもありません。

IPv6には、IPv4に存在したブロードキャストという仕組みがなく、IPv4でブロードキャストが担っていた役目をすべてマルチキャストが担うようになりました。そのため、マルチキャ

†1 RFC 1112：S.E. Deering, “Host extensions for IP multicasting”, 1989年8月

(cc) BY NC SA

ストはIPv6にとって不可欠な仕組みです。本章では、IPv6におけるマルチキャストの基本的な事項を整理します。

IPv6におけるマルチキャストアドレス設定上の注意

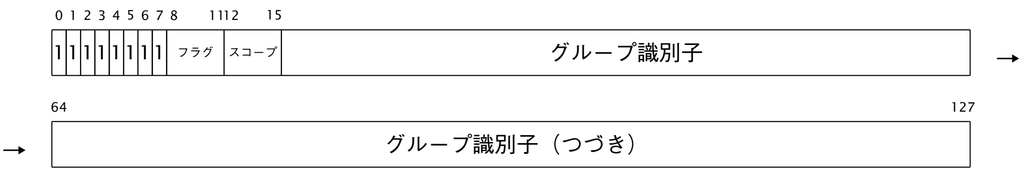
IPv6マルチキャストは、IPv4におけるブロードキャストの役割を含むので、ルータやファイアウォールなどでマルチキャストをフィルタしてしまうと通信ができなくなるおそれがあり注意が必要です。たとえば、ローカルセグメントでIPv4マルチキャストをフィルタするように設定されることがありますが、そのようなIPv4での設定をそのままIPv6に適用すると、DHCPv6などが利用できなくなってしまいます。

なお、同様の問題はICMPv6についてもいえます。ICMPv6については、RFC 4890[†2]で、IPv4とまったく同じフィルタ設定にすることで障害が発生する要因になる点が議論されています。

11.1　IPv6のマルチキャストアドレス

IPv6では、最初の8ビットがすべて1になるIPアドレスがマルチキャストアドレスです。IPv6アドレスのテキスト表記でいえば、`ff00::/8`がマルチキャストアドレスになります。

IPv6アドレスにおいて、マルチキャストであることを示す先頭の8ビットに続く4ビット（8ビットめ〜11ビットめの）はフラグとして使われます。さらに後ろに続く4ビット（先頭から12ビットめ〜15ビットめ）は、マルチキャストのスコープを表します。図11.1にIPv6マルチキャストアドレスの構造を示します。



▶ 図11.1　IPv6マルチキャストアドレス

図11.1の各フィールドは次のようになっています。

- スコープ（4ビット）

  マルチキャストスコープを表します。IPv6のマルチキャストスコープに関しては11.2節で説明します。

- フラグ（4ビット）

  4ビットのフラグは図11.2のように割り当てられています。

  8ビットめは必ず0となります。

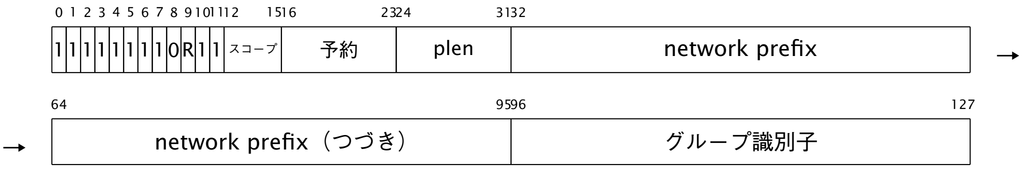
†2　RFC 4890：E. Davies, J. Mohacsi, “Recommendations for Filtering ICMPv6 Messages in Firewalls”, 2007年5月

	8	9	10	11	
···	0	R	P	T	···

▶ 図11.2　IPv6マルチキャストアドレスのフラグ

11ビットめのTは、既知の固定アドレスであるかどうかを示しています。IANAなどによって規定されている既知のマルチキャストアドレスであれば、この部分は0となります。一方で、固定的ではなく一時的に利用するためのマルチキャストアドレスでは、この部分が1となります。

10ビットめのPは、ユニキャストのネットワークプレフィックスを含むマルチキャストアドレスであることを示しています。Pが1の場合はTの値も1になります。Pが1の場合、IPv6マルチキャストアドレスは図11.3のようなフォーマットになります。



▶ 図11.3　Pフラグが1の場合のIPv6マルチキャストアドレス

マルチキャストでは、マルチキャストグループに参加しているノードが、そのグループを指すマルチキャストアドレス宛のトラフィックを受信します。複数の異なるアプリケーションで同じマルチキャストアドレスを同時に利用すると、不必要なトラフィックがネットワーク上を行き来してしまいます。そうしたマルチキャストアドレスの競合を回避するため、マルチキャストアドレスについては、割り振りや割り当ての方法が問題になります。マルチキャストアドレスの割り当てと割り振りに関する概要はRFC 6308[†3]でまとめられています。マルチキャストアドレスの自動割り当てを行う方法もあり、RFC 3306[†4]では、ユニキャストアドレスのネットワークプレフィックスを用いて他のネットワークで利用されているマルチキャストアドレスとの競合を防ぎつつマルチキャストアドレスを自動割り当てする方法が定義されています。

NOTE

11.5.3項で後述するように、マルチキャストには送信元を指定しないASM（Any-Source Multicast）と、送信元を指定するSSM（Source Specific Multicast）があり、マルチキャストアドレスの割り振りや割り当てが大きな意味を持つのはASMのほうです。SSMでは、マルチキャストグループと送信元が一体として考慮されるので、マルチキャストグループが同じであっても送信元が別であれば異なるトラフィックとして扱われます。そのためSSMでマルチキャストアドレスの競合が発生する可能性があるのは、送信元が同一になるノード内においてのみです。

†3　RFC 6308：P. Savola, “Overview of the Internet Multicast Addressing Architecture”, 2011年6月
†4　RFC 3306：B. Haberman,D. Thaler, “Unicast-Prefix-based IPv6 Multicast Addresses”, 2002年8月

図11.3のplenは、ユニキャストアドレスのプレフィックス長を表します。それに続くnetwork prefixフィールドには、Pフラグが1の場合にマルチキャストアドレス中で利用されるユニキャストのネットワークプレフィックスを指定します。

IPv6そのものの定義からはプレフィックス長の最大値が64ビットであると仮定できませんが、IPv6のアドレス体系を定義しているRFC 4291[†5]のSection 2.5.1に示されるインターフェース識別子に関する仕様に従って運用されている場合には、一般的なユニキャストアドレスのプレフィックス長の最大値は64ビットとなります。そういった背景もあり、このフィールドの長さは64ビットとされています（3.4.1項の/127についての説明も参照）。

その後に続くグループ識別子は、RFC 3307[†6]で規定された方法を用いて割り当てられます。

Rビットは、中継機器を使ってルータをまたぐマルチキャストを実現する場合に、その中継機器（ランデブーポイント、Rendezvous Point）のアドレスがマルチキャストアドレスに埋め込まれていることを表しています。Rビットが1の場合のマルチキャストアドレスの構造についてはRFC 3956[†7]を参照してください。

11.2　マルチキャストのスコープ

マルチキャストアドレスによってトラフィックが配送される範囲を、マルチキャストのスコープといいます。IPv6では、アドレス体系の一部として、宛先として指定するマルチキャストアドレスによって配送範囲を調整できる仕組みが最初から定義されています。

NOTE

マルチキャストのスコープは、3.3節で説明したIPv6アドレスの有効範囲を示すスコープとは異なるので注意してください。

IPv6マルチキャストのスコープとしては、以下のような種類が定義されています。なお、IANAのサイトでは、これらの各スコープごとにIPv6マルチキャストアドレスの割り当て一覧が公開されています[†8]。

- 1：Interface-Local（インターフェースローカルスコープ）
- 2：Link-Local（リンクローカルスコープ）
- 3：Realm-Local（レルムローカルスコープ）
- 4：Admin-Local（アドミンローカルスコープ）
- 5：Site-Local（サイトローカルスコープ）

---

†5　RFC 4291：R. Hinden,S. Deering, “IP Version 6 Addressing Architecture”, 2006年2月

†6　RFC 3307：B. Haberman, “Allocation Guidelines for IPv6 Multicast Addresses”, 2002年8月

†7　RFC 3956：P. Savola, B. Haberman, “Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address”, 2004年11月

†8　IPv6 Multicast Address Space Registry：
https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml

CC BY NC SA

- 8 ： Organization-Local（組織ローカルスコープ）
- E ： Global（グローバルスコープ）

1のインターフェースローカルスコープは、単一のインターフェース内でのみ利用可能なマルチキャストです。インターフェース内でのループバックでのみ受け取ることが可能です。

2のリンクローカルスコープは、単一リンク内（単一サブネット内）に対して配送されるマルチキャストです。

3のレルムローカルスコープは、RFC 7346[†9]で追加されたもので、リンクローカルスコープより範囲が広いものの、ネットワーク体系に依存するようなマルチキャストスコープです。RFC 7346では、ノード同士がメッシュでつながっている状況での利用例が記述されています。

4のアドミンローカルスコープは、管理される最小単位でのネットワークを表します。

5のサイトローカルスコープは、単一サイト内でのマルチキャストです。

8の組織ローカルスコープは、同一組織が保持する複数サイトをまたがったマルチキャストのスコープを表します。

Eのグローバルスコープは、グローバルなマルチキャストのスコープを表します。

IANAのアドレス割り当てにおいても、これらのスコープが関連します。たとえば、NTP（Network Time Protocol）で利用するマルチキャストアドレスは、IANAのサイトでは`ff0X:0:0:0:0:0:0:101`と記述されています。これにスコープを当てはめた実際のマルチキャストアドレスは以下のようになります。

- `ff01::101` ： 同一インターフェース内に存在するNTPサーバ
- `ff02::101` ： 同一サブネット内のNTPサーバ
- `ff04::101` ： アドミンローカルスコープでのNTPサーバ
- `ff05::101` ： サイト内のNTPサーバ
- `ff08::101` ： 組織内のNTPサーバ
- `ff0e::101` ： インターネット上のすべてのNTPサーバ

NOTE

IPv4では、1998年に発行されたRFC 2365[†10]でマルチキャストのスコープが定義されています。RFC 2365では、`239.0.0.0/8`が、Administratively scoped IPv4 multicast spaceとされています。この`239.0.0.0/8`を宛先IPv4アドレスとするマルチキャストは、グローバルなインターネットには配送されず、単一の運用管理体制下にあるネットワーク内でのみマルチキャストとして配送されます。なお、RFC 2365以前は、IPv4マルチキャストの配送範囲を制限するのに主にTTLが利用されていました。

---

†9 RFC 7346：R. Droms, “IPv6 Multicast Address Scopes”, 2014年8月
†10 RFC 2365：D. Meyer, “Administratively Scoped IP Multicast”, 1998年7月

CC BY NC SA

11.3　Solicited-Nodeマルチキャストアドレス

Solicited-NodeマルチキャストアドレスはRFC 4291で定義されており、ユニキャストもしくはエニーキャストに使うIPv6アドレスの下位24ビットを利用したマルチキャストアドレスです。Solicited-Nodeマルチキャストアドレスでは、上位104ビットを`ff02::1:ff00:0`とし、下位24ビットをユニキャストアドレスもしくはエニーキャストアドレスの下位24ビットとします。たとえば、`2001:db8::01:800:200e:8c6c`というIPv6アドレスに対応するSolicited-Nodeマルチキャストアドレスは`ff02::1:ff0e:8c6c`です。

IPv6を利用するすべてのノードは、ネットワークインターフェースに対してIPv6アドレスを設定するとき、そのネットワークインターフェースにおいてSolicited-Nodeマルチキャストアドレスに参加することが求められます。

Solicited-Nodeマルチキャストアドレスは、第6章で解説する近隣探索プロトコルで使われます。Solicited-Nodeマルチキャストアドレスを利用することによって、近隣探索プロトコルで行われるアドレス解決を行うときに、必要なノードに対してのみマルチキャストで関連するメッセージを届けることができます。IPv4では、アドレス解決には同一リンク内のすべてのノードに対するブロードキャストが利用されています。これがIPv6ではマルチキャストになるので、ネットワークやノードに対する負荷が軽減され、効率が良くなることが期待されています。

NOTE

IPv6アドレスの上位ビットだけが異なる複数のIPv6アドレスは同一のSolicited-Nodeマルチキャストアドレスになるので、マルチプレフィックスかつマルチアドレスの状態でノードがリンク層で参加しなければならないマルチキャストグループの数が減るという効果があります(RFC 4291およびRFC 4861[†11])。ただ、2017年に発行されたRFC 8064[†12]によって、固定されたインターフェース識別子の利用が非推奨となったので、参加するマルチキャストグループの数が減ることによる効果はあまりなくなっているかもしれません。

11.4　マルチキャストにおけるゾーン

RFC 4007[†13]では、IPv6アドレスのスコープに対応する範囲のことをゾーンと呼んでいます(3.6節参照)。ユニキャストおよびエニーキャストではリンクローカルスコープとグローバルスコープに対応する2種類のゾーンだけを使いますが、マルチキャストでは多様なゾーンが利用されます。

マルチキャストアドレスが複数のゾーンに同時に属する例を見てみましょう。図11.4のような、ネットワークインターフェースが4つあるノードがあるとします。このノードのネットワークインターフェース1は、ループバックリンクであり、物理的なデバイスではなく仮想的

[†11] RFC 4861：T. Narten,E. Nordmark,W. Simpson,H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, 2007年9月

[†12] RFC 8064：F. Gont,A. Cooper,D. Thaler,W. Liu, “Recommendation on Stable IPv6 Interface Identifiers”, 2017年2月

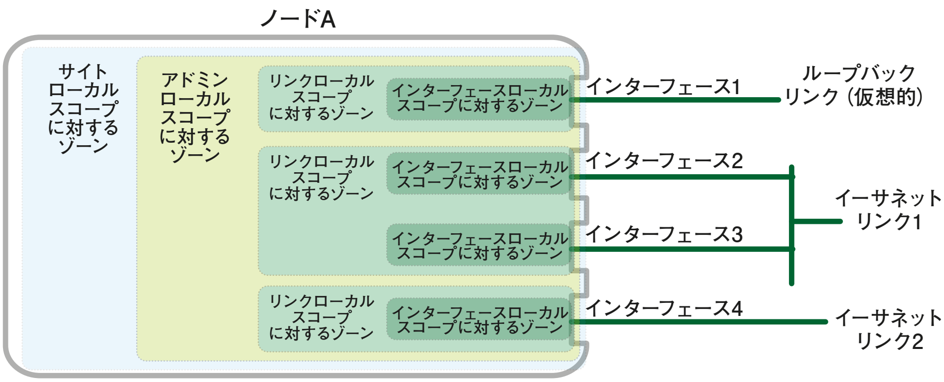
[†13] RFC 4007：S. Deering,B. Haberman,T. Jinmei,E. Nordmark,B. Zill, “IPv6 Scoped Address Architecture”, 2005年3月

なネットワークインターフェースとして構成されています。その仮想的なネットワークインターフェースにより、リンクローカルスコープに対応するゾーンが形成されています。また、それ単体でインターフェースローカルスコープに対応するゾーンを形成しています。

ネットワークインターフェース2と3も、それぞれが個別のインターフェースローカルスコープに対応するゾーンを形成しています。しかし、これらは同じリンク（イーサネットリンク1）に接続しており、リンクローカルスコープに対応するゾーンとしてはいずれも同じものに属しています。

ネットワークインターフェース4は、2および3とは別のリンク（イーサネットリンク2）に接続されています。やはりインターフェースローカルスコープに対応する単一のゾーンを形成しているほか、イーサネットリンク2に関係するリンクローカルスコープが形成するゾーンに、ノードA内のインターフェースとしては唯一属しています。

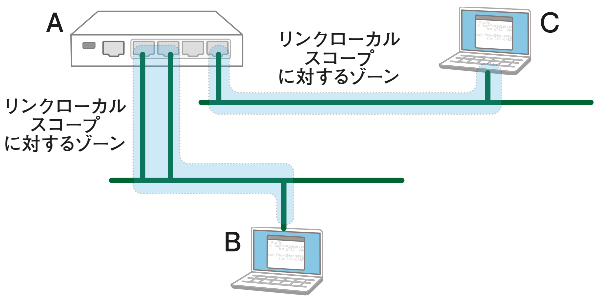
さらに、これらの全インターフェースは、11.2節で説明したマルチキャストのスコープに対してもゾーンを形成します。具体的には、アドミンローカルスコープに対応するゾーンに属するとともに、サイトローカルスコープに対応するゾーンにも属しています。



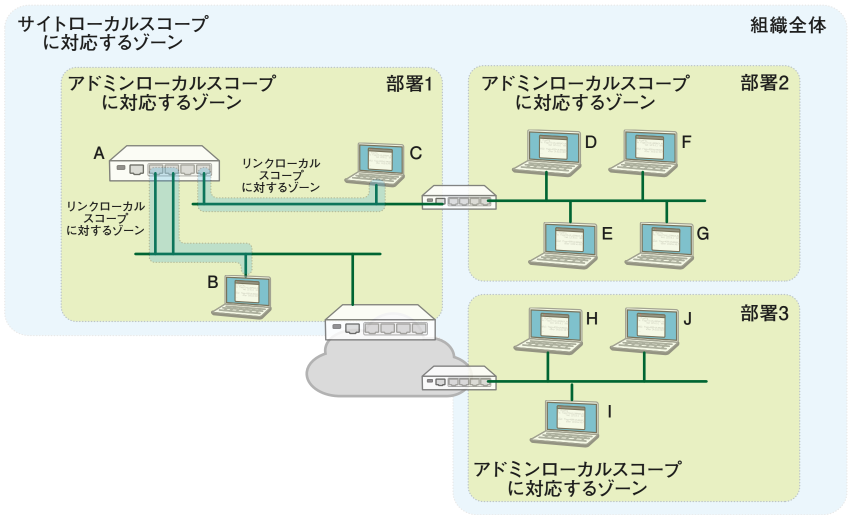
▶ 図11.4 ノードAの視点で見たマルチキャストの各スコープのゾーン

図11.4は、ノードA内という視点でゾーンを見たものです。今度は、これを一歩引いた視点で見てみましょう。まず、ノードAが属するリンクローカルスコープに対するゾーンは、たとえば図11.5のような範囲で形成されるでしょう。

次に、ノードAは組織のある部署が管理している範囲にある端末だとします。このとき、図11.6のように、部署1に属する端末でアドミンローカルスコープに対応するゾーンが形成できます。さらに、サイトローカルスコープに対応するゾーンを、組織全体のネットワークで形成されるものとして設定できるでしょう。このように、より大きなスコープに対応するゾーンは、より小さなスコープに対応するゾーンを内包する形で構成します。IPv6では、このようなゾーンの考え方に沿ってIPv6マルチキャストアドレスを設計し、マルチキャストトラフィックの配送範囲を制御できます。



▶図11.5 リンクローカルスコープに対応するゾーン



▶図11.6 アドミンローカルスコープとサイトローカルスコープに対応するゾーン

11.5 MLD（Multicast Listener Discovery）

マルチキャストの受信側は、マルチキャストグループへの参加を表明することで、そのグループ宛に送信されたパケットを受信できるようになります。IPv4では、マルチキャストグループへの参加や離脱を制御するために、IGMP（Internet Group Management Protocol）というIPプロトコルが利用されていました。IGMPは、インターネットの初期から存在していたプロトコルのひとつです。実際、IGMPのプロトコル番号は2です。

IPv6では、マルチキャストグループへの参加や離脱といったマルチキャストの制御に、ICMPv6のMLD（Multicast Listener Discovery）という機能を利用します。MLDはICMPv6の機能の一部であり、基本的な部分はICMPv6（第5章参照）をベースにしています。言い換えると、IPv6におけるマルチキャストの制御では、IPv4におけるIGMPのような独自のプロトコル番号を持つ独立したプロトコルは使いません。

本書執筆現在、MLDにはバージョン1とバージョン2があります。MLDv2は、MLDv1との互換性を維持しつつも、特定の送信元アドレスからのマルチキャストのみを対象としてグ

cc BY NC SA

ループに参加するSSM（11.5.3項参照）という機能が追加されたバージョンです。IPv4におけるIGMPにもIGMPv3という拡張がありますが、そのIPv6版がMLDv2といってもいいでしょう（表11.1）。実際、MLDv2を定義しているRFC 3810でも、MLDv2はIGMPv3プロトコルをIPv6のセマンティクスへと翻訳したものであると書かれています。

▶ 表11.1 IPv4におけるIGMPとIPv6におけるMLDの対応

IPv4	IPv6
IGMPv1	対応するバージョンは存在せず
IGMPv2	MLDv1
IGMPv3	MLDv2

11.5.1 MLDで利用するICMPv6メッセージ

MLDv1は、RFC 2710[†14]で定義されています。ここではRFC 2710で定義されているMLDの基本的なICMPv6メッセージ（MLDメッセージ）について説明します。

MLDメッセージには、Multicast Listener Query（タイプ130）、Multicast Listener Report（タイプ131）、Multicast Listener Done（タイプ132）の3種類があります。Multicast Listener Queryメッセージは、マルチキャスト受信者が存在しているかどうかを確認するために、ルータがリンクに対して定期的に送信するメッセージです。Multicast Listener Reportメッセージは、マルチキャストの受信者となることを表明するためにノードが利用するメッセージです。Multicast Listener Doneメッセージは、マルチキャストグループからの脱退を表明するために使われます。以下に、各MLDメッセージの使い方を概説します。

- Multicast Listener Queryメッセージ

  リンク上にマルチキャスト受信者が存在しているかどうかを確認するために、ルータはタイプ130のICMPv6メッセージを送信します。同一リンク上でMulticast Listener Queryを送信するルータは、基本的には1つだけです。このルータをQuerierと呼びます。ルータは、もし自分よりもIPv6アドレスの数値が小さいルータから送信されたMulticast Listener Queryメッセージを確認した場合、自分ではMulticast Listener Queryメッセージを送信しないルータ（Non-Querier）となります。

  Multicast Listener Queryメッセージには、General Queryと、Address-Specific Queryの2種類があります。General Queryは、Querierであるルータから定期的に、リンクスコープである全ノードアドレス（`ff02::1`）宛に送信されるメッセージです。General Queryを受信したノードは、自分が参加しているマルチキャストグループを、後述するMulticast Listener Reportメッセージでルータに伝えます。ルータが最初に起動するときには、リンク上に存在するすべてのマルチキャスト受信者を把握するために、General Queryを送信することが推奨されています。

[†14] RFC 2710：S. Deering, W. Fenner, B. Haberman, “Multicast Listener Discovery (MLD) for IPv6”, 1999年10月

CC BY NC SA

一方、Address-Specific Queryは、特定のマルチキャストグループに存在しているノードを知るためのメッセージです。そのため、Address-Specific Queryは、調べたいマルチキャストグループを指すマルチキャストアドレス宛に送信されます。

なお、ルータで考慮するのは各リンクにマルチキャストの受信者が存在しているのかどうかだけであり、マルチキャスト受信者の数は把握しません。

- Multicast Listener Reportメッセージ

マルチキャストグループに参加するノードは、タイプ131のICMPv6メッセージであるMulticast Listener Reportメッセージを送信します。ノードでは、ルータからのMulticast Listener Queryメッセージを受信すると、Queryに対応するマルチキャストアドレス宛にMulticast Listener Reportメッセージを返信します。ルータがMulticast Listener Reportメッセージを受け取り、そのリンクに該当するマルチキャストグループに参加しているノードがいることを確認したら、ルータからそのリンクに対してマルチキャストを送信するようになるわけです。

ただし、Multicast Listener Queryメッセージを受け取ったノードはすぐにMulticast Listener Reportメッセージを返信するわけではなく、ランダムな時間だけ待って返信します。その待ち時間に他のノードから該当するマルチキャストアドレス宛に送信されたMulticast Listener Reportを受け取った場合には、自分ではMulticast Listener Reportメッセージを送信しません。

Multicast Listener Reportメッセージは、Multicast Listener Queryメッセージに対する応答だけでなく、ノードが自発的に送信する場合もあります。RFC 2710では、そのリンク内で最初にマルチキャストを受信するノードになる可能性に備えて、マルチキャストグループに参加したときにノードがMulticast Listener Reportメッセージを送信すべきとされています。さらに、最初のMulticast Listener Reportメッセージに対して障害が発生する可能性を考慮し、間隔を空けて1つまたは2つを追加で送信することが推奨されています。

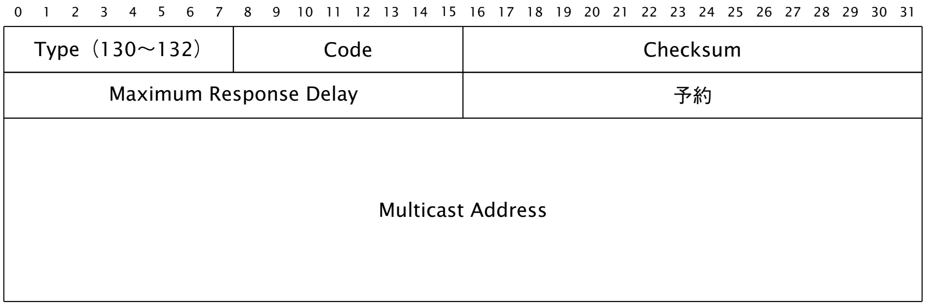
- Multicast Listener Doneメッセージ

マルチキャスト受信者は、マルチキャストグループから脱退するときにMulticast Listener Doneメッセージを送信します。Multicast Listener Doneメッセージを受け取ったQuerierは、そのマルチキャストグループに対するMulticast Listener Queryメッセージを送信します。このMulticast Listener Queryメッセージに対する応答としてMulticast Listener Reportメッセージを受け取れなかった場合は、そのリンク上に対象となるマルチキャストグループの受信ノードが存在しないと判断します。その判断をするためにMulticast Listener Reportメッセージの到着を待つ時間は、Multicast Listener DoneメッセージのMaximum Report Delayフィールドで指定します。

図11.7に、上記3種類のMLDメッセージの基本フォーマットを示します。

MLDメッセージを配送するIPv6パケットでは、IPv6ヘッダのHop Limitが1である必要があります。これは、MLDメッセージはリンク内で利用するためのものであり、ルータを越えないからです。また、MLDメッセージを配送するIPv6パケットの送信元IPv6アドレスはリンク

cc BY NC SA



▶ 図 11.7　MLD メッセージフォーマット

ローカルスコープとします。

MLD メッセージの先頭 4 オクテットは、ICMPv6 の基本ヘッダのフォーマットと同一です。MLD メッセージでは Code フィールドは使われておらず、すべて 0 に設定されます。

Maximum Response Delay フィールドは、Multicast Listener Query メッセージに対する Multicast Listener Report の応答の待ち時間を指定するフィールドで、Multicast Listener Query メッセージのみで利用されます。Multicast Listener Report メッセージおよび Multicast Listener Done メッセージでは、このフィールドはゼロに設定します。

Multicast Address フィールドには、各タイプの MLD メッセージで伝える必要があるマルチキャストアドレスを送信します。具体的には、Multicast Listener Report メッセージおよび Multicast Listener Done メッセージではノードが参加するマルチキャストグループのマルチキャストアドレスを指定します。Multicast Listener Query メッセージでは、Address-Specific Query の場合には調べたいマルチキャストアドレスを指定します。General Query の場合には、このフィールドはゼロに設定します。

■ Router Alert オプション

MLD パケットは、IPv6 拡張ヘッダの Hop-by-Hop オプションに Router Alert オプションを含めて送信されます。Router Alert オプションは、それを受け取ったルータが IP パケットを精査することを求めるもので、RFC 2711 で定義されています。

Router Alert オプションを含めることで、自分自身を宛先としない IPv6 パケットのうち中身の確認が必要なものをルータが識別しやすくなります。MLD では、通常のパケットを転送するためのルータの負荷を抑制するために、Router Alert オプションを使います。RSVP（Resource Reservation Protocol）、MRD（Multicast Router Discovery）、PGM（Pragmatic General Multicast）などのプロトコルでも、同じ目的で Router Alert オプションが利用されます。

Router Alert オプションのフォーマットを図 11.8 に示します。

Value フィールドは、どのような値を運んでいるかを 16 ビットの数値で示すもので、MLD の場合はゼロに設定します。Value フィールドで利用可能な値は IANA の Web サイトで公開さ

CC BY NC SA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	Value															

▶図11.8 Router Alertオプション

れています[†15]。

Router Alertオプションを悪用することで、途中経路上のルータに過度な負荷をかけてしまうといった攻撃が可能になります。そのため、Router AlertオプションをEnd-to-Endのインターネットで利用しないように求めるRFC 6398[†16]が発行され、これによりRFC 2711は更新されています。ただし、MLDを運ぶIPv6パケットの送信元アドレスがリンクローカルであるため、ルータを越えないことから、RFC 6398で懸念されている途中ルータに過負荷を与える攻撃には利用できません[†17]。

NOTE

IPv4にもRouter Alertオプションがあります。IPv4のRouter AlertオプションはRFC 2113[†18]で定義されており、RFC 2113はRFC 5350とRFC 6398によって更新されています。

11.5.2 MLDv2

MLDv2は、受信者が興味のある送信元を限定できるようにする機能をMLDv1に対して追加したものです。MLDv1と互換性がある設計になっており、RFC 3810[†19]で定義されています。

MLDv2は、MLDv1と同様の機能に加えて、マルチキャストのトラフィックの送信ノードを指定するためのINCLUDEフィルタとEXCLUDEフィルタを指定できるのが大きな特徴です。本書では、MLDv2の詳細には立ち入りません。詳細はRFC 3810を参照してください。

11.5.3 SSM

RFC 1112で定義されているマルチキャストは、ある特定のマルチキャストグループに対して参加したノードが送信元に関係なく対象となるマルチキャストグループのパケットを受け取るというものです。このような送信元が特定されないマルチキャストは、**ASM**（Any-Source Multicast）と呼ばれています。それに対して、マルチキャストグループと送信者の両方を指定できるのが**SSM**（Source Specific Multicast）です。

SSMは、RFC 4607[†20]で定義されています。IPv6マルチキャストにおけるSSMでは、xをスコープ識別子として、`ff3x::/32`というマルチキャストアドレスが利用されます[†21]。IPv6の

†15 IPv6 Router Alert Option Values：`https://www.iana.org/assignments/ipv6-routeralert-values/ipv6-routeralert-values.xhtml`
†16 RFC 6398：F. Le Faucheur, Ed., “IP Router Alert Considerations and Usage”, 2011年10月
†17 MLDv2のRFCであるRFC 3810は、Hop Limitが1であることをチェックすることも求めています。
†18 RFC 2113：D. Katz, “IP Router Alert Option”, 2011年10月
†19 RFC 3810：R. Vida, L. Costa, “Multicast Listener Discovery Version 2 (MLDv2) for IPv6”, 2004年6月
†20 RFC 4607：H. Holbrook, B. Cain, “Source-Specific Multicast for IP”, 2006年8月
†21 IPv4の場合は`232.0.0.0/8`が利用されます。

マルチキャストアドレスであることを示す先頭8ビットのffに続く3は、11.1節で説明したPフラグとTフラグが両方とも1であることを示しています。

MLDv2を利用してIPv6でSSMを実現するための仕様は、IPv4におけるIGMPv3を利用したSSMと合わせて、RFC 4604[†22]で定義されています。

11.5.4 Lightweight MLDv2

MLDv2の簡易版として、Lightweight-MLDv2（LW-MLDv2）と呼ばれるものがRFC 5790[†23]で定義されています。LW-MLDv2は、MLDv2のEXCLUDEフィルタを実装しないことによって、MLDv2の簡易版を実現するという仕様です。MLDv2ではSSMを実現するときなどにINCLUDEフィルタを利用しますが、EXCLUDEフィルタについては利用しない場合が多いので、EXCLUDEフィルタを必要としない簡易版として標準化されたのがRFC 5790です。

11.5.5 MLDとDADの問題

MLDv1が定義されているRFC 2710の仕様では、MLDメッセージの送信元IPv6アドレスとしてリンクローカルアドレスを使うことが要求されています。さらに、Multicast Listener Reportメッセージを送信する際にマルチキャストグループへの参加も要求されています。これらの仕様は、実は近隣探索プロトコルの仕様と競合します。

まず、近隣探索プロトコルによるアドレス自動設定（SLAAC）では、アドレスの重複がないかどうかをDADでチェックするために複数のマルチキャストグループに参加する必要がありますが、その時点ではまだ送信元IPv6アドレスが決定していません。この仮のIPv6アドレスを通信に利用することはできないので、MLDメッセージを送信する際の送信元IPv6アドレスとして利用可能なIPv6アドレスが存在しないという状況が発生してしまいます。そこで、適切な送信元IPv6アドレスが存在しない場合にMulticast Listener Reportメッセージを未定義アドレス（::）で送信できるようにするため、RFC 2710を上書きするRFC 3590[†24]が策定されました。

さらに、RFC 4862[†25]のSection 5.4.2では、11.7節で後述するMLDスヌーピングを行うスイッチによってSLAACにおけるDADが正常に動作しない可能性があることが指摘されています。MLDスヌーピング機能があるスイッチでは、Multicast Listener Reportメッセージが送信されたあとに、関連するマルチキャストパケットを該当するポートに対して配送するようになります。Multicast Listener Reportメッセージを送信するまでは、Solicited-Nodeマルチキャストアドレスに送信されるDADのためのパケットが受信できない可能性があるのです。実際、電源障害などに起因して複数のノードでネットワークインターフェースの同時設定が発生する

†22 RFC 4604：H. Holbrook, B. Cain, B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", 2006年8月

†23 RFC 5790：H. Liu, W. Cao, H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", 2010年2月

†24 RFC 3590：B. Haberman, "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", 2003年9月

†25 RFC 4862：S. Thomson, T. Narten, T. Jinmei, "IPv6 Stateless Address Autoconfiguration", 2007年9月

cc BY NC SA

場合など、同一リンク上に同時に多数のMulticast Listener Reportメッセージが送信されてしまい、スイッチにおけるMLDスヌーピングの処理が追いつかず、DADが正常に機能しない可能性があります。そのような状態を避けるため、RFC 4862では、ランダムな時間だけ待ってからSolicited-Nodeマルチキャストグループへ参加することが推奨されています。

11.6　ルータを越えるマルチキャスト

ネットワーク層でのマルチキャストでは、ルータを越えてマルチキャストグループに参加しているノードへとパケットを転送できる必要があります。そのための機能を持ったルータはマルチキャストルータと呼ばれています。マルチキャストルータは、必要に応じてIPパケットを複製しながら転送するだけでなく、マルチキャストパケットの経路を決める必要があります。

ネットワーク層でのマルチキャストでパケットの経路を決めるには、複雑な処理が必要になります。ユニキャストであれば特定の宛先に対する経路（次ホップ）が1つに定まりますが、マルチキャストでは複数のサブネットにパケットをコピーして送信する必要があり、一般に複数の次ホップが存在するからです。受信するノードが動的にマルチキャストグループに入ったり抜けたりできるので、配送先が頻繁に変わる可能性もあります。

マルチキャストルータでは、マルチキャストの次ホップを決定するために、マルチキャストグループを宛先とするパケットの配送経路を示す木構造（マルチキャスト配信ツリー）を構築します。そのために利用される仕組みとして、PIM（Protocol Independent Multicast）と呼ばれるものがあります。

PIMにはPIM-SM（Sparse Mode）とPIM-DM（Dense Mode）という2種類があります。PIM-SMでは、ランデブーポイント（RP）と呼ばれる機能を持ったルータを設定し、マルチキャストの各送信元はRPまでの配信ツリーを作成します。一方、サブネット内のノードからMLDで特定のマルチキャストグループへの参加を表明されたルータは、やはりRPに対し、配下のノードが当該のマルチキャストグループに参加していることをPIMで伝えます。こうして、送信側と受信側の双方からRPに向けてマルチキャスト配信ツリーが構築されます。RPから受信ノードへの配信ツリーはすべての送信ノードで共有することになるので、このような配信ツリーを共有ツリーと呼びます。PIM-SMについての詳細はRFC 7761[†26]を参照してください。

RPから受信ノードまでの配信ツリーを共有するPIM-SMに対し、各送信ノードから受信ノードまでの配信ツリーをすべて途中のルータで管理する方式がPIM-DMです。この場合の配信ツリーは、宛先だけでなく送信元についても経路表の要素とするので、送信元ツリーと呼ばれます。PIM-DMは、受信側のノードがあまり散らばっておらず、密集（dense）している場合に利用されます。PIM-DMについての詳細はRFC 3973[†27]を参照してください。

---

†26　RFC 7761：B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, L. Zheng, “Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)”, 2016年3月

†27　RFC 3973：A. Adams, J. Nicholas, W. Siadak, “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)”, 2005年1月

cc BY NC SA

11.7 MRD（Multicast Router Discovery）

リンク層が複数のスイッチで構成されている場合に、IPv6 マルチキャストを流すと、該当するマルチキャストグループに参加しているノードが接続されていないスイッチにもトラフィックが流れてしまう可能性があります。そのような状況で無駄なトラフィックを抑制するために、MLD のメッセージをリンク層レベルで覗き見（snoop）し、IPv6 マルチキャストを考慮してフォワーディングテーブルを設定する機能（MLD スヌーピング）がスイッチに実装されている場合があります。

リンク層のスイッチで MLD スヌーピングを利用するには、マルチキャストグループに参加するノードだけでなく、リンクにおけるマルチキャストルータの情報も必要になります。そこで、接続している環境でどのノードがマルチキャストパケットを転送するマルチキャストルータとして稼働しているのかを知るための仕組みが MRD（Multicast Router Discovery）として標準化されています。MRD は、RFC 4286[†28]で定義されています。

NOTE

MRD は IPv4 マルチキャストでも利用されます。IPv4 マルチキャストの場合、MLD スヌーピングに相当する機能は IGMP スヌーピングと呼ばれています。

IPv6 の MRD では、次の 3 種類の ICMPv6 メッセージを使うことで、マルチキャストのデータや MLD のメッセージを送信すべきルータをノードが発見できるようにします。

- Multicast Router Advertisement メッセージ（タイプ 151）
- Multicast Router Solicitation メッセージ（タイプ 152）
- Multicast Router Termination メッセージ（タイプ 153）

NOTE

IPv4 では、MRD は IGMP における Multicast Router Advertisement メッセージ（タイプ 0x30）、Multicast Router Solicitation メッセージ（タイプ 0x31）、Multicast Router Termination メッセージ（タイプ 0x32）として定義されています。

Multicast Router Advertisement メッセージは、マルチキャストルータであることを広告するために、マルチキャストルータのすべてのインターフェースから定期的に配信されます。ノードからの Multicast Router Solicitation メッセージへの応答として返送される場合もあります。Multicast Router Solicitation メッセージは、ノードがマルチキャストルータを探すときに利用されます。Multicast Router Termination メッセージは、あるインターフェースでマルチキャストルータとしての機能を停止するときに、接続しているリンクから送信されます。

MRD に利用する ICMPv6 メッセージは、Hop Limit が 1 のパケットによって送信されます（IPv4 の場合には TTL が 1 のパケット）。その際の IPv6 パケットとしての宛先アドレスは、リンクローカルスコープのマルチキャストアドレスのひとつとして IANA で割り当てられている All-Snoopers と呼ばれるアドレスとします。All-Snoopers は、IPv6 では `ff02::6a` です（IPv4

[†28] RFC 4286：B. Haberman, J. Martin, “Multicast Router Discovery”, 2005 年 12 月

CC BY NC SA

では224.0.0.106です)。

MLDスヌーピング対応スイッチを利用する場合の参考情報

RFC 4541[†29]では、MLDスヌーピング対応スイッチの実装に関して推奨される手法などが紹介されています。RFC 4541では、MLDスヌーピングを行うスイッチがマルチキャストグループへの参加を通知するMulticast Listener Reportメッセージを転送する際に、マルチキャストルータが接続されているポートに限定することなどが推奨されています。また、送信元IPv6アドレスが未定義アドレス(::)であるMLDメンバーシップレポートメッセージを破棄すべきではないといったことが述べられています。

RFC 4541はInformationalなRFCなので、あくまで実装に関して推奨される参考情報がまとめられているものです。とはいえ、MLDスヌーピングを行うスイッチを利用する場合にも有用な情報が含まれているので、MLDスヌーピングを利用する場合にはぜひ参照してください。

11.7.1　MRDで利用されるICMPv6メッセージ

Multicast Router Advertisementメッセージのフォーマットを図11.9に示します。

0–7	8–15	16–31
Type(151)	Ad. Interval	Checksum
Query Interval		Robustness Variable

▶ 図11.9　Multicast Router Advertisementメッセージ

Typeフィールドには、Multicast Router Advertisementメッセージであることを示す151を指定します。ICMPv6の基本ヘッダにおけるCodeフィールドに相当するAd. Intervalフィールドには、Multicast Router Solicitedメッセージへの応答ではない定期的なAdvertisementメッセージの送信間隔を秒単位で示します。Query Intervalフィールドには、MLDにおいてGeneral QueryのMulticast Listener Queryメッセージを送信する間隔を秒単位で指定します。MLDを利用していないインターフェースではゼロを設定します。Robustness Variableフィールドには、MLDにおいてリンク上でのパケットロス対策に利用される係数を指定します。

Multicast Router SolicitationメッセージとMulticast Router Terminationメッセージのパケットフォーマットは単純で、それぞれICMPv6の基本ヘッダのみの構造です(図11.10)。Typeフィールドは、Multicast Router Solicitationメッセージの場合は152、Multicast Router Terminationメッセージの場合は153です。ICMPv6の基本ヘッダにおけるCodeフィールドは0に設定します。

[†29] RFC 4541:M. Christensen, K. Kimball, F. Solensky, “Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches”, 2006年5月

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Type（152または153）	0	Checksum

▶ 図11.10　Multicast Router Solicitationメッセージと Multicast Router Terminationメッセージ

11.8　リンク層でのマルチキャストアドレス

マルチキャストは、ネットワーク層だけで実現される通信方法ではありません。リンク層のプロトコルでもマルチキャストの方法が規定されている場合があります。リンク層におけるマルチキャストは、ネットワーク層におけるマルチキャストと異なり、基本的には同一セグメント内のみが送信範囲です。なお、同じHUBやスイッチに接続されているかどうかにかかわらず、同じサブネットに存在するPC同士は同一セグメントです。

リンク層におけるマルチキャストパケットは、単純な仕組みでは全ポートにコピーして転送されます。ただし、ネットワーク層プロトコルの設定によって、送信される対象の機器が変わることもあります。IPv6では、MLDスヌーピング対応スイッチによって必要なポートに対してのみマルチキャストパケットを転送するようにできる場合があります。

11.8.1　IPv6アドレスからイーサネットにおけるマルチキャストアドレスを生成する

リンク層プロトコルであるイーサネットにもマルチキャストがあります。イーサネットでは、宛先がユニキャストの場合にはネットワークインターフェースに設定されているMACアドレスを利用しますが、宛先がIPv6マルチキャストの場合には、そのIPv6マルチキャストアドレスに対応するイーサネットアドレスが必要になります。

IPv6マルチキャストアドレスに対応するイーサネットアドレスを生成する方法は、RFC 2464[†30]で定義されています。RFC 2464では、図11.11のように、0x33　0x33に続けて、IPv6マルチキャストパケットの宛先アドレスのうちの13オクテットめから16オクテットめまでの値をイーサネットにおけるマルチキャストアドレスとして利用するとあります。たとえば、`ff02::1`というIPv6マルチキャストアドレスに対応するイーサネットアドレスは、`33:33:00:00:01`になります。

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
IPv6宛先アドレスの第13オクテット								IPv6宛先アドレスの第14オクテット							
IPv6宛先アドレスの第15オクテット								IPv6宛先アドレスの第16オクテット							

▶ 図11.11　IPv6マルチキャストアドレスに対応するイーサネットアドレス

ただし、そのIPv6マルチキャストの受信ノードが明確に1つであるとわかる場合には、リンク層において宛先アドレスをマルチキャストではなくユニキャストとしてもよいという拡張が

[†30] RFC 2464：M. Crawford, “Transmission of IPv6 Packets over Ethernet Networks”, 1998年12月

RFC 6085[†31]で行われています。それに伴い、IPv6 マルチキャストパケットのリンク層における宛先アドレスがユニキャストであるという理由で、IPv6 マルチキャストの受信ノードがパケットを破棄することが禁止されています。

†31 RFC 6085：S. Gundavelli, M. Townsley, O. Troan, W. Dec, “Address Mapping of IPv6 Multicast Packets on Ethernet”, 2011 年 1 月

BY NC SA

第12章

IPv6エニーキャスト

エニーキャスト（anycast）は、anyという英単語の意味のとおり、「複数のうちいずれか近くのもの」に到達する形態の通信です。IPでの通信におけるエニーキャストは、「まったく同じIPアドレスを持つ機器をインターネット上に複数存在させたうえで、経路を調整することでユーザが最寄りの機器へ到達できるようにする」という意味になります。IPv4とIPv6におけるエニーキャストの運用に関しては、RFC 4786[†1]（BCP 126）でまとめられています。

IPにおけるエニーキャストでは、ユニキャストIPアドレスを利用するので、IPアドレスを見ただけではエニーキャストで運用されているものかどうかわかりません。ただし、特定の用途のエニーキャストについては、IANAで予約されているIPアドレスがあります。それらのIPアドレスであれば、IPアドレスを見てエニーキャストであることがわかります。

IPv6では、エニーキャストがユニキャストとマルチキャストと並んで基本的な通信形態のひとつとしてプロトコルに含まれています。とはいえ、エニーキャストそのものはIPv4でも利用されており、そこでの運用がIPv6の提案当初から発展していることもあって、IPv6エニーキャストに関する仕様も提案当初と比べてかなり大きく変化しています。

12.1 IPv4におけるエニーキャスト

IPv4では、ユニキャストIPv4アドレスを利用したエニーキャストが行われています。IPv4エニーキャストの有名な運用例としては、ルートサーバ（root server/ルートネームサーバ/DNSルートサーバ）が挙げられます（RFC 3258[†2]）。世界各地で運用される13系統のルートサーバは、同じIPv4アドレスを持つ複数のサーバによって運用されています。

注意が必要な点として、かつては複数のDNSサーバを同じIPv4アドレスで運用する技術のことをエニーキャストとは呼ばず、「IPv4アドレスの共有」と表現していました。実際、2002年に発行されたRFC 3258では、「Anycast」ではなく「Shared Unicast Addresses」とされて

†1 RFC 4786：J. Abley, K. Lindqvist, “Operation of Anycast Services”, 2006年12月
†2 RFC 3258：T. Hardie, “Distributing Authoritative Name Servers via Shared Unicast Addresses”, 2002年4月

CC BY NC SA

います。エニーキャストの歴史をまとめたRFC 7094[†3]のSection 2.1では、エニーキャストという表現が避けられていた理由について、IPv4では後述するon-linkエニーキャストを実現しているわけではない点を挙げています。

NOTE

22.1節で紹介する6to4という技術もIPv4エニーキャストを利用していました。ただし、6to4にはさまざまな問題があるので、いまでは「使うべきではない」とされている技術ではあります。

12.2　IPv6のエニーキャスト

IPv6では、プロトコルの仕様でエニーキャストに対応しています。しかし、ユニキャストやマルチキャストと比べると、エニーキャストにIPv6の根幹を成すような利用方法があるわけではありません。実際、ルーティング技術を利用してルータを越える遠隔地を対象とするエニーキャスト（off-linkエニーキャスト）は、IPv4とIPv6とで事実上同じものといえます。

一方、同一リンク内エニーキャスト（on-linkエニーキャスト）は、IPv6でのみ標準化されています。つまり、IPv6エニーキャストがIPv4エニーキャストと異なるのは、同一リンク上にIPv6エニーキャストアドレスを持つインターフェースが複数存在する場合です。IPv6では、IPv6エニーキャストが最初から設計に含まれているので、同一リンク上に同じIPv6アドレスが存在できます。これは、IPv6エニーキャストアドレスに対してはIPv6の基本機能である同一リンク上の衝突検出が行われないからです（7.6節参照）。

12.2.1　IPv6エニーキャストアドレスの仕様変更

IPv6におけるエニーキャストの仕様は、昔と今とで異なります。その代表が、IPv6エニーキャストを利用可能な場面です。

初期の仕様では、IPv6エニーキャストアドレスはホストに対して設定してはならず、ルータに対してのみ設定が可能でした。また、IPv6エニーキャストアドレスは宛先アドレスとしてのみ利用可能であり、送信元アドレスとして利用することは禁止されていました（RFC 1884[†4]のSection 2.5、RFC 2373[†5]のSection 2.6、RFC 3513[†6]のSection 2.6をそれぞれ参照）。

しかし、これらの仕様は、2006年に発行されたRFC 4291[†7]で廃止されました。このRFC 4291はRFC 3513を上書き廃止するものです。この仕様変更の結果、現在ではIPv6エニーキャストを送信元アドレスとして利用可能です。また、IPv6エニーキャストをルータだけではなくサーバでも利用できるようになっています。

この仕様変更によるもう1つの大きな変化として、IPv6エニーキャストアドレスを持つノー

†3　RFC 7094：D. McPherson, D. Oran, D. Thaler, E. Osterweil, “Architectural Considerations of IP Anycast”, 2014年1月

†4　RFC 1884：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 1995年12月

†5　RFC 2373：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 1998年7月

†6　RFC 3513：R. Hinden, S. Deering, “Internet Protocol Version 6 (IPv6) Addressing Architecture”, 2003年4月

†7　RFC 4291：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 2006年2月

CC BY NC SA

ドとTCPを利用した通信が行えるようになりました。TCPでは、送信元IPアドレス、宛先IPアドレス、送信元ポート番号、宛先ポート番号の4つの情報を利用してTCPコネクションを識別します。昔のIPv6エニーキャストでは、送信元IPv6アドレスとしてIPv6エニーキャストアドレスの利用が禁止されていたので、TCP SYNパケットが送信される宛先IPアドレスと、それに対する返信としてのTCP SYN/ACKパケットの送信元IPアドレスが異なることになり、TCPコネクションが張れないという制約がありました。RFC 4291によって送信元IPアドレスとしてIPv6エニーキャストアドレスも利用可能になったことから、IPv6エニーキャストでのTCPの利用が可能になりました。

12.2.2 IPv6エニーキャストを利用するノード側での設定

IPv6エニーキャストアドレスは、IPv6ユニキャストアドレスと同じIPv6アドレス空間を利用するので、マルチキャストアドレスのように機械的には見分けられません。しかし、IPv6エニーキャストとIPv6ユニキャストには異なる点もあるので、IPv6エニーキャストアドレスを宛先とするパケットの処理には特別な設定が必要です。このため、IPv6エニーキャストアドレスを利用するノードでは、あるIPv6アドレスがエニーキャストであることを認識できる必要があります。

IPv6におけるエニーキャストとユニキャストの大きな違いは、エニーキャストアドレスに対してはDAD（Duplicate Address Detection）が行われないことです（RFC 4862[†8]のSection 5.4参照）。DADは、同一セグメント上に同じIPv6アドレスが設定されたネットワークインターフェースが存在しているかどうかを確認するためのものなので、これが行われないということは、同一セグメント上に同じIPv6エニーキャストアドレスが同時に存在できることになります。なお、IPv4のエニーキャストでは、同一セグメント上に同じIPv4アドレスが設定されたネットワークインターフェースが存在すると、システムのネットワークインターフェースに設定されているIPv4アドレスと同じものをARPで受け取り、結果としてIPアドレスの重複についての警告がシステムから出る場合があります。

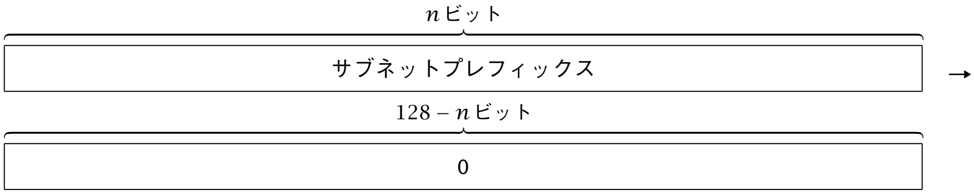
12.3 IPv6サブネットルータエニーキャストアドレス

RFC 4291のSection 2.6.1では、サブネットに接続されたルータのネットワークインターフェースに対するエニーキャスト用として、**サブネットルータエニーキャストアドレス**というものが定義されています。サブネットルータエニーキャストアドレスとして予約されているのは、インターフェース識別子がすべて0となるIPv6アドレスです（図12.1）。すべてのルータは、サブネットに接続されたネットワークインターフェースに対し、サブネットルータエニーキャストアドレスを設定する必要があります。

---

†8 RFC 4862：S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration”, 2007年9月

CC BY NC SA



▶ 図12.1　IPv6サブネットルータエニーキャストアドレス

12.4　ユニキャストとしての利用を避けるべきIPv6アドレス

RFC 2526[†9]では、各サブネットにおける上位128個のIPv6アドレスを、サブネットエニーキャストアドレスとして予約しています。さらに、RFC 5375[†10]のAppendix B.2.5.2では、これらのIPv6アドレスをユニキャストアドレスとして利用することを避けるように推奨しています。

RFC 5453[†11]では、サブネットエニーキャストアドレスのために、`fdff:ffff:ffff:ff80` ～ `fdff:ffff:ffff:ffff`のインターフェース識別子をIANAで予約しています。RFC 5453では、すべてが0となるサブネットルータエニーキャストのためのインターフェース識別子も同時に予約しています。

12.5　エニーキャストの注意点

12.5.1　同じエニーキャストアドレスで運営されている別ノードにパケットが配送されることがある

RFC 7094では、エニーキャスト運用時の注意点として、エニーキャストで運用されているノードとの通信中に相手が変わる可能性があると紹介されています。つまり、ある特定のエニーキャストアドレスに対して複数のパケットを送信したとき、そのすべてが同じノードに向けて配送されるとは限りません。実際、RFC 7094では、エニーキャストで運営されるサービスに向けての通信では単一の要求パケットに対して単一の応答とするのが安全であるとしています。

あるアドレス宛の複数のパケットが別々のノードに到達することが問題になる例として、ステートフルで再送などを行うTCPのようなプロトコルの利用が挙げられます。RFC 7094によれば、TCPにおけるエニーキャストの利用は推奨されないという解釈も可能です。

RFC 7094では、Path MTU Discoveryをエニーキャストアドレスに対して行う場合にも、同様の問題の可能性があると説明されています。あるアドレスまでのPath MTUを調べるための複数のパケットが、同じエニーキャストアドレスを持つ異なる相手に届いている可能性がある

---

†9　RFC 2526：D. Johnson, S. Deering, “Reserved IPv6 Subnet Anycast Addresses”, 1999年3月

†10　RFC 5375：G. Van de Velde, C. Popoviciu, T. Chown, O. Bonness, C. Hahn, “IPv6 Unicast Address Assignment Considerations”, 2008年12月

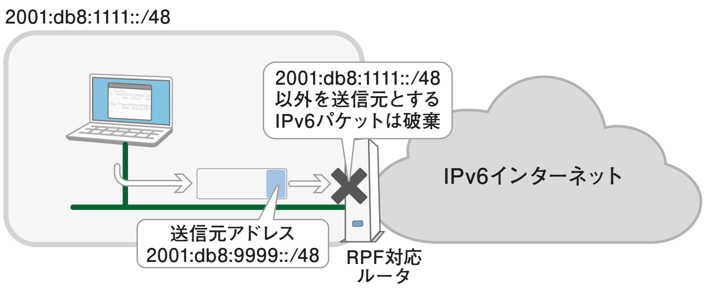
†11　RFC 5453：S. Krishnan, “Reserved IPv6 Interface Identifiers”, 2009年2月

からです。

エニーキャストアドレスが送信元となっている場合にも注意が必要です。エニーキャストアドレスから送信されたパケットに対して期待される応答が、同じエニーキャストアドレスで運営されている別の相手に向けて配送された結果、返ってこない可能性があるからです。

12.6 IPv6 エニーキャストと BCP 38

2000年に発行されたRFC 2827[†12]（BCP 38）では、各自が管理するネットワークで偽装されたパケットを出してDDoS攻撃に加担してしまうことを防ぐため、**RPF**（Reverse Path Forwarding）という手法が提案されています[†13]。これは、各パケットの送信元へのユニキャストの経路を参照しつつ、「そのパケットがその方向からくることが正しいかどうか」を確認するというものです。このRPFにより、IPv6エニーキャストアドレスを送信元アドレスとしたパケットが破棄される可能性があります（図12.2）。



▶ 図 12.2　BCP 38

同様の問題はマルチホーム環境においても発生します。マルチホーム環境におけるRPFについては、RFC 3704[†14]で議論されています。RFC 3704は、BCP 38を補うものとして2004年に発行されたものですが、その中でマルチホーム環境も考慮したゆるい（Loose）RPFなどが提案されています。IPv6エニーキャストアドレスについても、このマルチホーム環境での対策と同じ考え方での運用が可能です。

---

†12　RFC 2827：P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", 2000年5月

†13　マルチキャストのRPFと明確に区別するために、**uRPF**（unicast Reverse Path Forwarding）と呼ばれることもあります。

†14　RFC 3704：F. Baker, P. Savola, "Ingress Filtering for Multihomed Networks", 2004年3月

[書籍版では空白のページです]

BY NC SA

第13章

IPv6におけるマルチアドレスとマルチプレフィックス

IPv6はIPv4と異なり、1つのネットワークインターフェースに対して複数のIPv6アドレスを設定可能です。本書では、1つのネットワークインターフェースに対して複数のIPv6アドレスを設定されている状態や、設定可能であることを**マルチアドレス**と表現します。また、複数のプレフィックスが1つのネットワークインターフェースに対して設定されることを**マルチプレフィックス**と呼びます。

IPv6のマルチアドレスとマルチプレフィックスに伴う課題については、近隣探索プロトコルに関連して、6.6節でもon-linkとoff-linkにまつわる議論を紹介しました。本章では、さらに以下の3つのトピックについて解説します。

- デフォルトIPv6アドレスの選択
- マルチプレフィックスによるマルチホームの問題
- IPv6サイトリナンバリング

デフォルトIPv6アドレス選択は、通信にかかわるエンドノードが複数のIPv6アドレスからどれをどのように選択して通信するかを判断するときのデフォルト値に関する話です。これはあくまでデフォルトであり、それ以外に何らかの方法が存在しないときに最終的に選択されるものです。

場合によっては、デフォルトIPv6アドレス選択だけでは通信に使うべきIPv6アドレスを選択できない場合もあります。たとえば、複数のグローバルユニキャストアドレスが上位ネットワークから提供されてマルチプレフィックスになるマルチホーム環境の場合です。マルチプレフィックスによるマルチホーム環境でどのような問題があるについては、13.2節で解説します。

13.3節では、マルチプレフィックスが利用可能であるというIPv6の特徴を生かして新旧両方のプレフィックスを並行してネットワーク内で運用しつつ、ネットワークの完全停止を避けながらサイトリナンバリングを実施する際の課題について扱います。

CC BY NC SA

NOTE

IPv6のマルチアドレスとマルチプレフィックスに伴う課題には未解決な点も多く残っています。本書執筆時点においては「課題がまとめられているだけ」という内容も少なくないことに注意してください。

13.1 IPv6アドレスの選択

IPv6では、1つのネットワークインターフェースに対し、スコープが異なる複数のユニキャストアドレスが設定される可能性があります。また、ネットワークインターフェースの識別子に基づいて決まるアドレスのほかに、Temporary IPv6アドレス（匿名IPv6アドレス）が利用される場合もあります。さらに、SLAACで生成され、DADが完了して有効になったアドレスには、有効期間が完全に切れる前の「利用が推奨されない状態」があります。この状態のアドレスは、deprecated（非推奨、160ページ参照）ではありますが、インターフェースに設定されていて利用は可能です。

IPv6対応ノードは、ある特定のネットワークインターフェースからパケットを送信する際に、そのネットワークインターフェースに設定されている複数のIPv6アドレスのうち1つを送信元IPv6アドレスとしてパケットに設定する必要があります。宛先アドレスに関しても、特定の名前に対応する複数のアドレスが存在する場合もあるので、何らかの優先順が必要です。どのIPv6アドレスを宛先IPv6アドレスや送信元IPv6アドレスとして選択するか、その方針がノードごとにバラバラな状態では、ネットワークの管理やアプリケーション開発に支障があります。そこで、IPv6アドレスを選択するデフォルトの方法が、RFC 6724[†1]として定義されています。

RFC 6724で想定されているのは、「通信を開始するアプリケーションが`getaddrinfo()`などのAPIを利用して、特定の名前で表現される相手との通信で利用する宛先アドレスを取得し、その宛先にパケットを送信する」という状況です。`getaddrinfo()`などのAPIは、IPv4アドレスを含む複数の宛先IPアドレスの候補をリストとして返す可能性があります。アプリケーションでは、それらが適切な順番にソートされているものとして、そのリストで先頭にくるものを使って`connect()`や`sendto()`などのシステムコールにより通信を開始します。そこでRFC 6724では、`getaddrinfo()`などのAPIで返される宛先アドレスのリストをソートするためのルールが定義されています。

宛先IPアドレスを決めるのがアプリケーションである一方で、送信元アドレスについては、宛先に応じた複数の候補から、OSカーネルなどが提供するネットワーク層で選択することが多くなります（アプリケーションが`bind()`などを利用して送信元IPアドレスを設定する場合もあります）。そこでRFC 6724では、アプリケーションが送信元IPv6アドレスを指定しない場合に、ネットワーク層の実装で適切な送信元IPv6アドレスを選択するためのルールも定義されています。

RFC 6724で定義されている宛先IPアドレス選択手法と送信元IPv6アドレス選択手法は、あ

---

†1 RFC 6724：D. Thaler, R. Draves, A. Matsumoto, T. Chown, “Default Address Selection for Internet Protocol Version 6 (IPv6)”, 2012年9月

CC BY NC SA

くまでもデフォルトを定めたものであり、明示的に設定された選択方法が別にあればそちらが利用されます。また、上位層による判断を上書きするものでもありません。上位層で選択の優先順位などが別途設定されている場合には、そちらが優先されます。実際、RFC 6724では、明示的に選択の優先順位などを設定する方法も定義されています。

13.1.1 送信元IPv6アドレス選択のルール

あるアドレスを宛先とするIPv6パケットに、どの送信元IPv6アドレスを設定するかは、いくつかの候補から選択します。送信元IPv6アドレスの候補となるのは、その宛先へIPv6パケットを送信するときに使うインターフェース（ルータの場合はパケットを転送することになるすべてのインターフェース）に設定されたユニキャストアドレスです。

送信元IPv6アドレスの候補から、実際にパケットに設定する送信元IPv6アドレスを1つ選択するアルゴリズムは、RFC 6724では8つのルールとして定義されています。各ルールは、候補となる2つのアドレスを比較するときの基準として定義されています。上位のルールに照らして比較したときに2つの候補が同順になった場合には、下位のルールを参照していきます。多くのプログラミング言語におけるswitch case文のようなものだと考えればよいでしょう。

ただし、アプリケーションで`bind()`システムコールなどを利用して送信元IPv6アドレスを選択する必要がある場合もあるので、これら8つのルールだけで送信元IPv6アドレスが選択されるわけではありません。アプリケーションで明示的な選択を行わない場合に送信元IPv6アドレスを選択する際の判断基準が、この8つのルールだといえます。

- ルール1：同じアドレスを優先する
  宛先アドレスと同じ送信元アドレスであれば、それを選択する
- ルール2：適切なスコープを選択する
  より小さいスコープを選択する
- ルール3：非推奨のアドレスは避ける
  SLAACで有効となったアドレスが2つあり、片方が最初の有効期間を過ぎてdeprecatedになっている場合は、そうでないほうを選択する
- ルール4：ホームアドレスを優先する
  モバイル端末に、通常利用するアドレスであるホームアドレス（Home Address）と、移動先で一時的に利用するアドレスである気付アドレス（Care of Address）が同時に設定されていたら、ホームアドレスを選択する
- ルール5：送出されるネットワークインターフェースのアドレスを優先する
  パケットの送信に使うネットワークインターフェースに設定されているアドレスを選択する。次ホップとなるルータから広告されたプレフィックスにマッチするアドレスがあれば、それを選択する
- ルール6：ポリシーテーブルでラベルがマッチするものを優先する
  後述するポリシーテーブルで、宛先アドレスのラベルと一致するラベルを持つアドレスがあれば、それを選択する

- ルール7：Temporary IPv6アドレスを優先する
  パブリックなアドレスよりもTemporary IPv6アドレスを選択する（ただし、このルールについては、アプリケーションによる指定で逆の意味にできるように実装すること）
- ルール8：宛先アドレスとより一致するものを優先する
  プレフィックスが宛先アドレスと一致する部分が長いアドレスを選択する。通信の質を考慮するなど、より適切な送信元IPv6アドレスを選択する手段がある場合は、その方法を優先してよい

13.1.2　宛先アドレス選択のルール

宛先アドレスの選択については、優先順を定めるリストをソートするための基準として、10のルールがRFC 6724に示されています。送信元アドレスの選択ではIPv6アドレスだけが対象でしたが、宛先アドレス選択に使うリストにはIPv4アドレスも含まれます。このとき、IPv4アドレスは、IPv4-Mapped IPv6アドレスとして表現しなければなりません。

送信元IPv6アドレスに対する8つのルールと同様に、下位のルールは上位のルールに対する同点決着のための方式として定義されています。一般的なアプリケーションでは、`getaddrinfo()`などによって得られる結果を順番に試していくので、リストで上位にあるアドレスが選択されやすくなります。

- ルール1：利用できない宛先は避ける
  到達不能である場合や、そのアドレスを宛先とした場合に選択される送信元アドレスがない場合には、他のアドレスを宛先IPv6アドレスとして優先する。あるアドレスが到達不能であると判断する例としては、未接続のネットワークインターフェースを経由しなければ到達できない宛先である場合や、近隣不到達性検知（6.3.5項）を継続して受けている場合が考えられる
- ルール2：送信元アドレスとスコープが一致するものを優先する
  あるアドレスを宛先IPv6アドレスとして選択した場合に、そのスコープが「送信元IPv6アドレスとして選択されるアドレスのスコープ」と同じになるなら、そのアドレスを宛先IPv6アドレスとして選択する
- ルール3：非推奨のアドレスは避ける
  あるアドレスを宛先IPv6アドレスとして選択した場合に、送信元IPv6アドレスとして選択されるアドレスがdeprecatedであれば、他のアドレスを宛先IPv6アドレスとして優先する
- ルール4：ホームアドレスを優先する
  あるアドレスを宛先IPv6アドレスとして選択した場合に、送信元IPv6アドレスとして選択されるアドレスがホームアドレスかつ気付アドレスである場合は、そのアドレスを宛先IPv6アドレスとして選択する。そうでなければ、送信元IPv6アドレスとして選択されるアドレスがホームアドレスであるほうを宛先IPv6アドレスとして選択する

- **ルール 5：ポリシーテーブルでラベルがマッチするものを優先する**
  後述するポリシーテーブルで、送信元 IPv6 アドレスのラベルと一致するラベルを持つアドレスがあれば、それを宛先 IPv6 アドレスとして選択する
- **ルール 6：ポリシーテーブルで優先度が高いものを優先する**
  後述するポリシーテーブルで、より高い優先度のものを宛先 IPv6 アドレスとして選択する
- **ルール 7：カプセル化で到達できる宛先は避ける**
  IPv4 にカプセル化される IPv6 通信などよりも、IPv6 のまま転送されて到達できるアドレスを宛先 IPv6 アドレスとして選択する
- **ルール 8：スコープが小さいほうを優先する**
  より小さいスコープのアドレスを宛先 IPv6 アドレスとして選択する
- **ルール 9：送信元 IPv6 アドレスとより一致するものを優先する**
  プレフィックスが、選択される送信元 IPv6 アドレスと一致する部分が長いほうのアドレスを、宛先 IPv6 アドレスとして選択する
- **ルール 10：リストの掲載順に従う**
  ルール 9 までで比較できない場合は、並べ替えをせず、現状のリストの掲載順で上位のものを選択する

RFC 6724 では、上記ルール 9 と 10 に関しては、他の方法がある場合には実装依存としてもよいとされています。

13.1.3 ポリシーテーブル

ポリシーテーブル（Policy table）は、デフォルトの IP アドレス選択のためのルールで利用できる、優先度（Precedence）およびラベル（Label）を設定するための仕組みです。優先度とラベルは、アドレスプレフィックスに対する値として設定します。

ポリシーテーブルにおける優先度は、宛先アドレス候補のリストを並べ替える際に利用される可能性があります。優先度が高いプレフィックスのほうが、低いプレフィックスよりも、リストで上位に並べられやすくなります。

ポリシーテーブルにおけるラベルは、対象となる IP アドレスを分類するためのものです。宛先アドレスの選択時には、送信元アドレスと同じラベルが付いたプレフィックスのほうが選択されやすくなります。送信元アドレスの選択時には、宛先アドレスと同じラベルが付いたプレフィックスのほうが選択されやすくなります。

RFC 6724 ではポリシーテーブルを設定可能とすることが推奨されていますが、設定可能でない実装や、設定前のデフォルトとして、表 13.1 のような内容のポリシーテーブルを推奨値としています。

cc BY NC SA

▶ 表13.1　RFC 6724で推奨されているデフォルトのポリシーテーブル

プレフィックス	優先度	ラベル
::1/128	50	0
::/0	40	1
::ffff:0:0/96	35	4
2002::/16	30	2
2001::/32	5	5
fc00::/7	3	13
::/96	1	3
fec0::/10	1	11
3ffe::/16	1	12

NOTE

glibcでは、RFC 6724が定義しているポリシーテーブルを実現する方法として、gai.confという設定ファイルがあります（15.5節参照）。

ポリシーテーブルにIPv4の優先度を設定するときは、IPv4アドレスの表現として、IPv4-Mapped IPv6アドレスが利用されます。たとえば、IPv4全体を示す場合には::ffff:0:0/96、192.0.2.0/24を示す場合には::ffff:192.0.2.0/120というIPv4-Mapped IPv6アドレスを利用します。IPv4アドレスに対するスコープとしては、IPv4自動設定アドレス169.254.0.0/16（RFC 3927[†2]）とIPv4ループバックアドレス127.0.0.0/8（RFC 1918[†3]のChapter 4）はリンクローカルスコープに、プライベートIPv4アドレスとISP Shared IPv4アドレス（19.4.3項）を含むその他のIPv4アドレスはすべてグローバルスコープに分類されています。

13.1.4　旧RFC 3484との違い

デフォルトIPv6アドレス選択のRFCとしては、2003年に発行されたRFC 3484[†4]がありました。このRFC 3484を2012年9月に置き換えたのがRFC 6724です。

RFC 3484とRFC 6724の主な違いは以下のとおりです。

- Temporary IPv6アドレス（匿名アドレス）が存在する場合は優先するようにルールが変更された
- IPv6サイトローカルアドレスがRFC 3879[†5]によって2004年に廃止されたことに伴い、RFC 6724では、設定例においてIPv6サイトローカルアドレスであるfec0::/10の優先度が最も低い値に変更された

†2　RFC 3927：S. Cheshire, B. Aboba, E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses”, 2005年5月

†3　RFC 1918：Y. Rekhter,B. Moskowitz,D. Karrenberg,G. J. de Groot,E. Lear, “Address Allocation for Private Internets”, 1996年2月

†4　RFC 3484：R. Draves, “Default Address Selection for Internet Protocol version 6 (IPv6)”, 2003年2月

†5　RFC 3879：C. Huitema, B. Carpenter, “Deprecating Site Local Addresses”, 2004年9月

- RFC 3484より後に定義されたULA（RFC 4193[†6]）に対する扱いがRFC 6724で追加された
- 6to4（22.1節参照）用のIPv6アドレスである2002::/32がIPv4よりも低い優先度になった。この変更により、将来の実装では「6to4を経由することによってIPv6で通信するほうがIPv4よりも通信品質が著しく悪くなる」という状態を回避しやすくなった
- DNSロードバランシングが機能しなくなる問題への対処として、ルールの一部が変更された
- グローバルアドレスが設定されているがIPv4でしかインターネットへの接続性がない状況への対応が追加された

13.2 マルチプレフィックスによるマルチホームの問題

ホストもしくはネットワークに対して上流の接続が複数存在する状態を**マルチホーム**と呼びます。IPv6では、1つのネットワークインターフェースに対して複数のIPv6アドレスを設定可能です。そのため、上流の接続を提供するネットワークから個別にネットワークプレフィックスを割り当て、マルチプレフィックス環境を利用することにより、マルチホームを実現できます。

とはいえ、ネットワークインターフェースに設定された複数のIPv6アドレスのうちどれを送信元アドレスとして使うかや、次ホップとなるルータの決め方、利用するDNSサーバによっては、通信の経路が変わったり、最悪の場合には通信ができなくなることもありえます。こうした問題は、NAT、もしくはIPv6のネットワークプレフィックスを変換するNPT（Network Prefix Translation）を導入することで解決できる場合もあります。

IPv4では、ルーティングプロトコルであるBGPを使うなどして、こうした問題を回避してマルチホームを実現する手段もありました。ただし、BGPを使う場合には、AS番号とIPアドレスの割り当てを受ける必要があり、それらの番号資源を維持管理することなどを含めて運用コストがかかります。そのような理由もあり、IPv4で小規模なネットワークでマルチホームを実現する場合にNATが利用される事例が多くあります。

このように、IPv4においてマルチホームを実現する主な手段はBGPかNATでした。IPv6はIPv4と異なり、1つのネットワークインターフェースに複数のIPv6アドレスを設定できるので、マルチホームを実現する手段としてマルチプレフィックスによる運用という選択肢が増えたということです。

本節では、IPv6におけるマルチプレフィックス環境を利用したマルチホームに伴う問題と、それらが発生する要因、そしてNATなどを導入することなく問題を回避してマルチホームを実現する手法を紹介します。

NOTE

IETFのhomenetワーキンググループでも、小規模なネットワークにおいてIPv6でマルチホーム環境をどのように実現すべきかという議論が行われています。

[†6] RFC 4193：R. Hinden, B. Haberman, “Unique Local IPv6 Unicast Addresses”, 2005年10月

(cc) BY NC SA

13.2.1 IPv6におけるマルチホームの3つのシナリオ

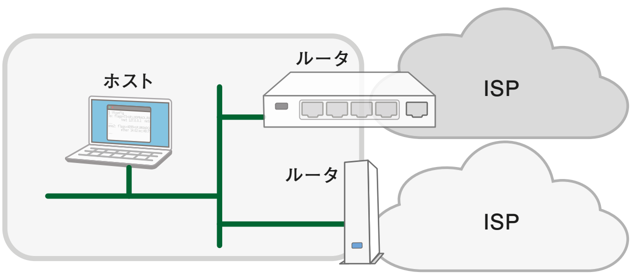
IPv6におけるマルチホーム環境で発生する問題に関しては、RFC 7157[†7]とRFC 7078[†8]でまとめられています。RFC 7157は、問題点と考えられる対策などがまとめられたInformationalなRFCです。

RFC 7157では、IPv6のマルチホームで問題が発生するシナリオとして、3つのシナリオが示されています。

■ シナリオ1

シナリオ1では、ホストが接続された単一のリンクに、2台以上のルータが存在しています（図13.1）。2台のルータは別々のサービスプロバイダに接続されており、それぞれサービスプロバイダから提供されたIPアドレスと再帰問い合わせ用DNSサーバに基づいて動作しています。リンク上にあるホストには、両方のルータからネットワークプレフィックスと再帰問い合わせ用DNSサーバがそれぞれ広告されます。したがって、このホストでは複数のネットワークプレフィックスと再帰問い合わせ用DNSサーバに関する情報を受け取ります。

複数のISPと契約し、個別のISPごとにルータを用意するような場合は、このシナリオ1に該当します。



▶ 図13.1　マルチホームのシナリオ1

■ シナリオ2

シナリオ2では、ホストが接続された1台のルータが、複数の上流ネットワークと接続しています（図13.2）。複数の上流ネットワークからは、それぞれネットワークプレフィックスと再帰問い合わせ用のDNSサーバが広告されます。ホストが接続するルータは、再帰問い合わせ用のDNSサーバの情報をすべてホストに伝える場合もあれば、簡易キャッシュDNSサーバとして稼働する場合もあります。

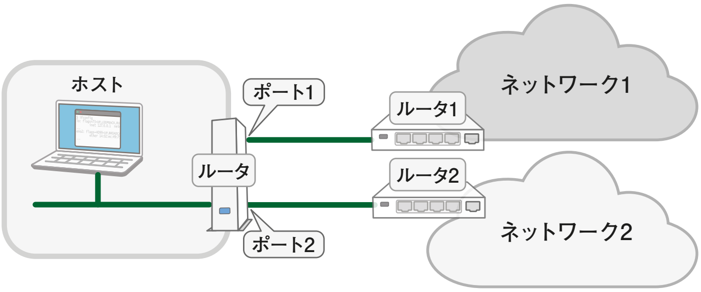
シナリオ2の例としては、インターネット接続と同時にVPN接続ができるように設定されたルータを運用している場合が挙げられます。

---

†7　RFC 7157：O. Troan, D. Miles, S. Matsushima, T. Okimoto, D. Wing, “IPv6 Multihoming without Network Address Translation”, 2014年3月

†8　RFC 7078：A. Matsumoto, T. Fujisaki, T. Chown, “Distributing Address Selection Policy Using DHCPv6”, 2014年1月

cc BY NC SA

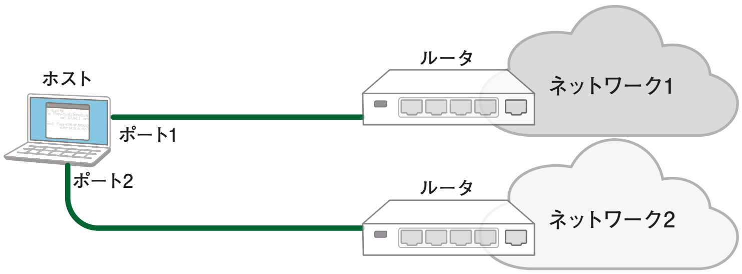


▶ 図 13.2　マルチホームのシナリオ 2

■ シナリオ 3

シナリオ3では、ホストが複数のネットワークインターフェースを持ち、それぞれのネットワークインターフェースが、別々のサービスプロバイダと通信するルータと接続されています（図 13.3）。ホストは、それぞれのサービスプロバイダから、ネットワークプレフィックスと再帰問い合わせ用のDNSサーバを広告されます。

たとえば、モバイル Wi-Fi と 3G 回線を同時に利用可能な機器や、複数の ISP からのインターネット接続サービスを同時に利用しているホストなどが、このシナリオ3と同じ構成になります。



▶ 図 13.3　マルチホームのシナリオ 3

13.2.2　マルチプレフィックスによるマルチホーム環境で発生する問題

マルチホームで接続しているホストが通信するとき、上流ネットワークにとっては存在しないはずのネットワークプレフィックスからのパケットが届いているように見えることがあります。これは、上流ネットワークが下流ネットワークとして認識しているネットワークプレフィックスではない、別の IPv6 アドレスを送信元 IPv6 アドレスとするパケットが、上流ネットワーク宛に送信されてしまうためです。この影響により、経路の途中で意図せず IPv6 パケットが破棄されてしまう可能性があります。

- DDoS 攻撃などの発信源になることを防ぐために、各パケットの送信元へのユニキャストの経路を参照しつつ「そのパケットがその方向からくることが正しいかどうか」を確認す

(cc) BY NC SA

る手法について、RFC 2827[†9]（BCP 38）で検討されている（12.6節参照）。この手法を実施しているマルチホーム環境では、送信元IPv6アドレスと経由するネットワークが一致してない場合にパケットが破棄されてしまう可能性がある
- 過去の通信内容に応じて通過可能なパケットを判断する、ステートフルなファイアウォールが途中経路上に存在している場合がある。マルチホームで接続しているホストからのパケットが、そこで破棄されてしまう可能性がある

13.1節で解説したRFC 6724によるデフォルトIPv6アドレス選択は、パケットの送信元IPv6アドレスを選択するためのデフォルトの方法であり、上記のような問題に対処するための要素のひとつといえます。しかし、デフォルトIPv6アドレス選択によっては、シナリオ1と2においてホストから送信するパケットの送信元IPv6アドレスを決定できません。RFC 6724では、複数の有効なグローバルアドレスから最適なものを選ぶ方法が規定されていないからです。複数のISPからグローバルアドレスのネットワークプレフィックスを割り当てられた状態などでは、RFC 6724によるデフォルト送信元IPv6アドレス選択によって送信元IPv6アドレスを決定することはできません。シナリオ3の場合は、パケットを送出するネットワークインターフェースに設定されているIPv6アドレスを送信元IPv6アドレスとすればよいので、送出するネットワークインターフェースが決まれば送信元IPv6アドレスを正しく設定可能です。

デフォルトIPv6アドレス選択の限界だけでなく、次ホップ選択で問題が生じることもありえます。たとえば、シナリオ1とシナリオ3の環境ではどちらか片方のルータがデフォルトルータとされ、場合によってはもう片方のルータに対してパケットが送出されない可能性があります。特にシナリオ3では、ホスト側で細かく経路を設定しなければ、片方だけにトラフィックが偏ってしまう可能性があります。

再帰問い合わせ用のDNSサーバの選択も問題です。シナリオ1、2、3のいずれの場合でも、再帰問い合わせのためのDNSサーバの選択が課題になる可能性があります。そのネットワークにおいてのみ利用可能であるようなローカルな名前空間があり、そのためのDNSサーバが存在している場合、再帰問い合わせのためのDNSサーバの選択と、名前検索の結果に対する通信で利用される送信元IPv6アドレスや経路が一致しないと、通信ができない場合があります。

このように、マルチホームにはいくつかの課題があります。RFC 7157では、これらの問題が発生する要因を以下の3つにまとめて掘り下げるとともに、考えられる解決策を提案しています。

- 間違った送信元IPv6アドレスが選択されてしまう
- 間違った次ホップが選択されてしまう
- 間違った再帰問い合わせ用DNSサーバが選択されてしまう

---

†9　RFC 2827：P. Ferguson, D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, 2000年5月

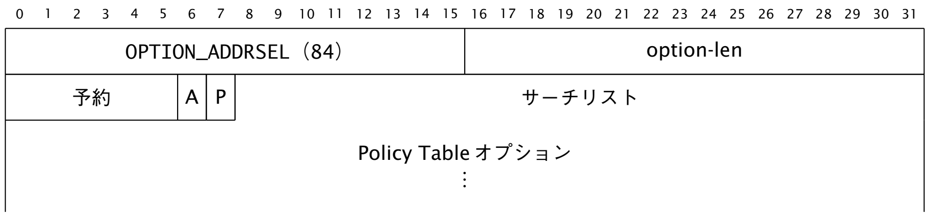
CC BY NC SA

13.2.3 送信元IPv6アドレスの選択における問題

シナリオ3ではパケットの送出ネットワークインターフェースが別々なので、それに応じて送信元IPv6アドレスを決定すれば、次の13.2.4項で説明するような次ホップ選択への対策により問題を回避できます。シナリオ1と2では、複数のIPv6アドレスが設定されているネットワークインターフェースから次ホップとしてのルータは1つなので、送信元IPv6アドレスに何を選択するかによって通信ができるかできないかが変わる可能性があります。

シナリオ1と2に対する解決策として、RFC 7157で提案されているのは、デフォルトIPv6アドレス選択における変数やポリシーテーブルの上書きに必要な情報をDHCPv6で配布するという手法です。そのためのDHCPv6オプションとして、RFC 7078ではAddress Selectionオプションが定義されています。

RFC 7078で定義されているAddress Selectionオプションのフォーマットを図13.4に示します。



▶ 図13.4 Address Selection オプション

Address Selectionオプションのオプションコードは84です。オプションのサイズを示すoption-lenには、6ビットの予約フィールド、AフラグとPフラグ、それに可変長のPolicy Tableオプションの合計の長さをオクテット単位で示します。

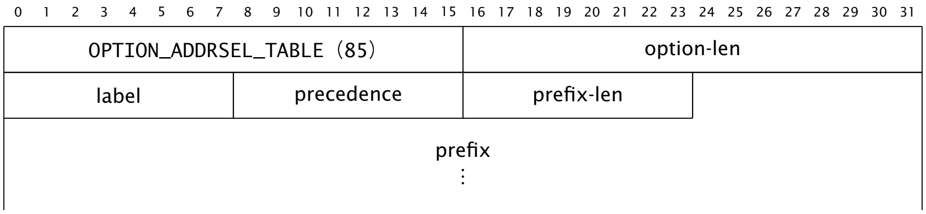
Aフラグは、RFC 6724のSection 2.1にあるAutomatic Row Additionフラグを設定するためのものです。このフラグが1に設定されている場合、クライアントは自動的にポリシーテーブルに項目を追加できます。このフラグが0である場合、DHCPv6のAddress SelectionオプションのPolicy Tableオプションに含まれているポリシーテーブルは意味をなしません。

Pフラグは、RFC 6724のSection 5にあるPrivacy Preferenceフラグを設定するためのものです。この値が1に設定されている場合、送信元IPv6アドレスとしてTemporary IPv6アドレスが優先されます。この値が0に設定されている場合、Temporary IPv6アドレスではなく、パブリックなアドレスが優先されます。

Address Selectionオプションに含まれるPolicy Tableオプションのフォーマットを図13.5に示します。

Policy Tableオプションのオプションコードは85です。option-lenフィールドには、label、precedence、prefix-len、prefixのフィールドのサイズの合計値をオクテット単位で示します。labelとprecedenceは、それぞれポリシーテーブルにおけるラベルと優先度を指定します。

prefix-lenフィールドには、ネットワークプレフィックス長を示す0から128の値を指定します。prefixフィールドには実際のネットワークプレフィックスを指定します。ネットワーク



▶図13.5　Policy Tableオプション

プレフィックス長が8の倍数でない場合、8ビット境界になるまで0でパディングされます。prefixフィールドは可変長で、0オクテットから16オクテットの間になります。

13.2.4　次ホップの選択

次ホップに複数の選択肢がある環境では、どのルータを経由して宛先までパケットを送信するのかをホストが決定する必要があります。たとえば、シナリオ1や3の環境で、ネットワーク1とネットワーク2の間に接続性がない場合、ネットワーク2に対するパケットをルータ1に対して送信してしまうと、そのパケットは宛先に届きません。このような場合、ホストは経路情報を考慮したうえで次ホップを選択する必要があります。

マルチプレフィックス環境のホストにおける最初のホップ選択については、2016年に、Standards TrackのRFC 8028[†10]が発行されています。RFC 8028では、デフォルトIPv6アドレス選択によって送信元が決定したあとに次ホップを選択する前提なので、次ホップ選択は送信元IPv6アドレスに応じて行うものとしています。具体的には、送信元IPv6アドレスをRouter Advertisementメッセージで広告したルータを次ホップとして選択するとしています。

13.2.5　キャッシュDNSサーバ選択

RFC 7157では、キャッシュDNSサーバの選択に対し、複数の再帰問い合わせ用DNSサーバに対して同時に問い合わせを行う方法と、どれか1つのDNSサーバを選択して問い合わせを行う方法が説明されています。しかし、これだけでは、DNSのクエリにおける送信元IPv6アドレスに応じて返信内容を変えるように設定されたDNSサーバがある場合など、目的のサーバのIPアドレスを取得できない場合もありえます。そこでRFC 7157では、DNSサーバを選択するためのポリシーをDHCPv6によって配布する仕様を定めたRFC 6731を紹介しています。

13.2.6　マルチホーム環境のためのIPv6 NAT

RFC 7157が前提としているのは、IPv4の小規模ネットワークにおいてマルチホームを実現するためにNAT（NAPT）が利用されてきた状況です。プライベートIPv4アドレスで運用されたネットワークでは、ユーザが利用するホストではなく、NAT機器が複数の上流ネットワークとの接続を仲介することで、次ホップ、送信元IPアドレス、DNSサーバを選択するのが一般的

†10　RFC 8028：F. Baker,B. Carpenter, “First-Hop Router Selection by Hosts in a Multi-Prefix Network”, 2016年11月

でした。

IPv6でも、NATを利用することで、ユーザ環境がマルチプレフィックスになることによる問題を防げます。しかし、RFC 7157で議論されているのは、あくまでもIPv6 NATを利用せずに問題を回避する場合の要素や対策です。RFC 7157の背景にはIPv6 NATを避けるという動機があるのです。

とはいえ、RFC 7157では、マルチプレフィックス環境でのマルチホームにおける問題に対する根本的な解決策が実現するまでの移行期においては、IPv6 NATが解決策になりうる環境があるともしています。実際、NTT NGNでは、マルチプレフィックス問題の解決策として導入されているIPv6 PPPoEでIPv6 NATを利用しています。これについては付録Aで改めて扱います。

NOTE

RFC 7157は、NTT NGNでのマルチプレフィックス問題について根本的な対策を考察していたNTTグループの社員が中心になって書かれたものです。マルチプレフィックス問題は、NTTフレッツだけの問題ではなく、RFC 7157に挙げられている3つのシナリオで必ず直面する問題です。RFC 7157の背景には、最初に問題に直面した経験を共有し、全世界で解決の方向に持っていきたいという動機があるのです。

NTT NGNによって発生するマルチプレフィックス問題は、IPv6 IPoEとIPv6 PPPoEという2種類の方法で解決されています。いずれも、ユーザが利用するネットワークをマルチプレフィックスにしないという手法での解決です。IPv6におけるマルチプレフィックス問題に対する解決策として当時実現可能だったのが、マルチプレフィックスにしないという方法だったともいえるでしょう。

13.3 IPv6サイトリナンバリング

IPv4では、あるネットワークで使われるIPアドレス全体を変更するような作業（サイトリナンバリング）を実施するとき、IPアドレスを切り替える作業として、そのネットワークを一時的に切り離したり停止したりする必要があります。「ネットワークで使われているIPv4アドレスの変更」というメンテナンスのために、ネットワークがダウンするわけです。これに対しIPv6では、そのような作業を実施する際にも「ネットワークを停止せずに運用を続けながら徐々に変更していく」という考え方があります。

IPv6は1つのネットワークインターフェースに対して複数のIPv6アドレスを設定できる仕様になっています。その特徴を活用して、あるサイト内で利用しているIPv6アドレスを切り替える際に、新旧のネットワークプレフィックスを両方とも運用することでスムーズな移行を実現できるようにしようというのが**IPv6サイトリナンバリング**です。

IPv6サイトリナンバリングはIPv6の基本的な仕様においても考慮されています。たとえば、近隣探索やIPv6アドレスの自動設定の仕様にはIPv6アドレスの有効期間という概念が組み込まれています。これはサイトリナンバリングを考慮したものだといえます。

とはいえ、本書執筆段階ではIPv6サイトリナンバリングにはさまざまな課題があり、簡単に実行できるとはいえない状況です。実際、IPv6サイトリナンバリングに関しては複数のInformationalなRFCだけが存在し、正式な方法を定義するといった位置づけのBCP（Best

cc BY NC SA

Current Practice）やStandards Trackがありません。下記に、IPv6サイトリナンバリングに関する各種のRFCとその役割を要約します。

- RFC 4192[†11]：ある特定のタイミングでネットワークを停止して一気にネットワークのIPv6アドレスを変更するようなことをしない前提で、ネットワークのIPv6を振り直す手順や、その際の課題が解説されています。
- RFC 5887[†12]："Renumbering Still Needs Work"（「リナンバリングには、まだ課題が多い」）と題されていることからわかるように、リナンバリングに関連する各種課題が紹介されています。
- RFC 6866[†13]：SLAACやDHCPv6などの動的なIPv6アドレス設定によらず、静的にIPv6アドレスが設定されているホストにおけるリナンバリングの問題がまとめられています。
- RFC 6879[†14]：企業内ネットワークやISPなど、大規模なエンタープライズネットワークにおけるリナンバリングを対象として、リナンバリングのシナリオ、課題、方法が解説されています。小規模なホームネットワークについては対象外としています。
- RFC 7010[†15]：IPv6サイトリナンバリングが抱える課題に関連するプロトコルなどの紹介と同時に、RFC 5887、RFC 6866、RFC 6879などで指摘されている課題の概要が紹介されています。IPv6サイトリナンバリングの全体像を把握するには、このRFCから読み進めるのがよいでしょう。

いずれのRFCでも、事前に計画されたIPv6サイトリナンバリングについて主に論じられています。不測の事態によって急にサイト全体のIPv6アドレスをリナンバリングするようなケースに関しては範疇外です。

上記のRFCのほかにも、ルータのリナンバリングに関するガイドとして、やはりInformationalなRFC 2072[†16]が1997年に発行されています。ただし、RFC 2072で提起されている問題の多くはCIDR（Classless Inter-Domain Routing）の導入開始に関連するものであり、すでに内容が古くなっています。RFC 2072はIPv4とIPv6の両方を対象としていますが、IPv6に関するリナンバリングに関しては、RFC 4192がRFC 2072を更新しています。

NOTE

Standards TrackのRFCであるRFC 2894[†17]では、ルータのリナンバリングを行うためのICMPv6フォーマットが定義されています。しかし、RFC 7010には"we are not aware of real network deployment"（実際のネットワークで導入されている例を知らない）とあるので、定義はされたものの誰も実装しなかったプロトコルとみなされているようです。

---

†11　RFC 4192：F. Baker, E. Lear, R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", 2005年9月

†12　RFC 5887：B. Carpenter, R. Atkinson, H. Flinck, "Renumbering Still Needs Work", 2010年5月

†13　RFC 6866：B. Carpenter, S. Jiang, "Problem Statement for Renumbering IPv6 Hosts with Static Addresses in Enterprise Networks", 2013年2月

†14　RFC 6879：S. Jiang, B. Liu, B. Carpenter, "IPv6 Enterprise Network Renumbering Scenarios, Considerations, and Methods", 2013年2月

†15　RFC 7010：B. Liu, S. Jiang, B. Carpenter, S. Venaas, W. George, "IPv6 Site Renumbering Gap Analysis", 2013年9月

†16　RFC 2072：H. Berkowitz, "Router Renumbering Guide", 1997年1月

13.3.1 RFC 4192で紹介されているリナンバリング手順

RFC 4192では、IPv6サイトリナンバリングを以下の8段階の手順で解説しています。

- 1. 初期段階

  ルーティング、ネットワークインターフェースに設定されたアドレス、フィルタ設定、ファイアウォール、その他システムで、古いネットワークプレフィックスが利用されている状態です。

- 2. リナンバリングプロセスの準備

  リナンバリングの最初のステップは、新しいネットワークプレフィックスと、そのネットワークプレフィックスのための逆引きゾーンの確保です。そのうえで、ネットワークに存在するすべてのリンクに対してサブのネットワークプレフィックスを割り当てる必要があります。この割り当ては、ネットワークに存在する機器に対する設定よりも前に実施します。実際にリナンバリングを始める前に、DNSサーバに登録されているレコードのTTLを短くしたり、DHCPクライアントからDHCPサーバへの問い合わせの頻度を多くしたりするといった再設定も必要です。

- 3. 新しいプレフィックスのための設定

  ルータやスイッチなどのネットワーク機器は新しいネットワークプレフィックスで稼働を開始しますが、新しいネットワークプレフィックスを利用した通信はまだ行わない状態です。IPv6では、同時に複数のIPアドレスが利用可能なので、古いネットワークプレフィックスと並行して新しいネットワークプレフィックスを利用したネットワーク運用が可能です。ファイアウォールに設定されているフィルタなどについても、この段階で設定を追加し、実運用に向けた準備を進めます。新しいネットワークプレフィックスでネットワークが正しく稼働していることを確認するためのテストも、この段階で実施します。

- 4. 新しいホストアドレスの追加

  新しいネットワークプレフィックスがネットワークで正常に稼働したら、ホストに対して新しいネットワークプレフィックスでのIPv6アドレスを追加します。この段階では、古いネットワークプレフィックスでも運用を続けます。ホストに対して新しいネットワークプレフィックスでのIPv6アドレスが追加されたら、DNSに登録されている情報も更新していきます。

- 5. 両方のプレフィックスの安定運用

  新しいネットワークプレフィックスでの設定がネットワークに追加され、それが安定するまでに十分な時間が経過した段階です。RFC 4192では、ネットワーク内のすべてのホストに対し新旧両方のプレフィックスでのアドレスが設定されている状態を「安定運用」と表現しています。

- 6. 古いプレフィックスから新しいプレフィックスへの移行

  新しいネットワークプレフィックスが完全に導入され、テストが終了し、安定運用に

---

†17 RFC 2894：M. Crawford, “Router Renumbering for IPv6”, 2000年8月

cc BY NC SA

至ったら、ホスト、ルータ、スイッチなどのネットワーク機器で新しいネットワークプレフィックスを利用し始めます。DNSサーバに設定された古いネットワークプレフィックスを新しいものに変更するなど、新しい環境へと徐々に移行していきます。

- 7. 古いプレフィックスの廃止
  古いネットワークプレフィックスを利用したすべてのセッションが終了し、古いネットワークプレフィックスへの依存がなくなった状態です。古いプレフィックスに関連するすべての設定を削除できます。
- 8. 最終段階：新しいプレフィックスの安定運用
  利用しているネットワークプレフィックスがすべて新しいものだけになった状態です。

13.3.2 IPv6サイトリナンバリングが抱える課題

サイト全体のサービスを特定のタイミングで停止せずにIPv6サイトリナンバリングを行う手順には、実現にあたってさまざまな困難があることが知られています。RFC 4192のIntroduction（序論）では、プロイセン王国の軍人であったカール・フォン・クラウゼヴィッツによる『戦争論』の悲観的な文章を紹介することで、IPv6サイトリナンバリングの難しさを強調しています。

> Everything is very simple in war, but the simplest thing is difficult. These difficulties accumulate and produce a friction, which no man can imagine exactly who has not seen war.... So in war, through the influence of an 'infinity of petty circumstances' which cannot properly be described on paper, things disappoint us and we fall short of the mark.
>
> （参考訳）戦争においてすべてのことは単純であるが、最も単純なことは難しい。困難は積み重なり、摩擦を発生させる。戦争を見たことがないものには想像できない摩擦を……。そのため戦争においては、紙面には適切に記せない「無限にある小規模な状況」の影響によって我々は失望し、目標を逃してしまう。

さらに、セキュリティ上の懸念に関して述べているRFC 4192のSection 5の最初の文章では、「リナンバリングのプロセスは理論上はわかりやすいが、実践では難しく、かつ危険である」と書かれています。設定ミスや攻撃によってサイト全体に対して大きな影響を与えてしまう可能性があるのです。

IPv6サイトリナンバリングには、セキュリティ上の問題以外にも、さまざまな課題があります。以下、RFC 5887で扱われているホストに関連する話題を中心に、IPv6サイトリナンバリングが抱える課題をいくつか紹介します。

- DNSに関連する問題として、ユーザが自分でDNSサーバを管理していない環境などでサイトリナンバリングが発生する場合に、DNSサーバに登録されている情報を更新できない可能性があることなどが紹介されています。
- トランスポート層では、送信元IPアドレスと宛先IPアドレス、それにポート番号を利用す

cc BY NC SA

る場合があります。たとえばTCPでは、一度通信が開始すると、通信開始時に利用したIPアドレスなどを変更することはできません。そのため、サイトリナンバリングにおいて古いIPアドレス設定が削除されるときに、影響を受けるセッションが発生する可能性があります。

- アプリケーションが一度利用したIPアドレスをキャッシュしている場合、サイトリナンバリングが行われることによって通信ができなくなる可能性が考えられます。ソフトウェアのライセンスを管理するためにIPアドレスを利用しているアプリケーションも考えられます。また、IPアドレスをURLやHTTPクッキーで利用していたり、アプリケーションのためのプロキシとしてIPアドレス情報が管理されている場合もあります。
- BSDソケットの実装では、DHCPv6（第8章参照）やSLAAC（第7章参照）などで通知されるIPアドレスの生存期間をアプリケーションが取得できる手段がありません。そのため、そのIPアドレスがいつまで利用可能であるのかをアプリケーションが感知できず、利用をやめるべき古いプレフィックスのIPアドレスを使い続けてしまう可能性があります。
- 固定IPアドレスによって運用されているノードが存在する場合、そのノードに関連する設定がさまざまなネットワーク機器に対して手動で行われているでしょう。そうした設定の更新については別途考える必要があるので、ネットワークの保守運用のための停止期間を発生させずにサイトリナンバリングを行うことが難しい場合もあります。固定IPv6アドレスで運用されているノードの対処に関する問題についてはRFC 6866でまとめられています。

NOTE

固定IPアドレスは手動設定によるIPアドレスであるとは限りません。たとえば、ステートフルDHCPv6（8.6節参照）によって自動的に固定IPアドレスを設定することもできます。

cc BY NC SA

[書籍版では空白のページです]

第14章

IPv6とセキュリティ

この章では、IPv6に関連するセキュリティについていくつか紹介します。

14.1 IPv6はIPv4よりもセキュアというわけではない

エンタープライズ環境におけるIPv6のデプロイについて紹介しているInformationalなRFC 7381[†1]に、「IPv6はIPv4よりもセキュアというわけではない（IPv6 Is No More Secure Than IPv4）」という章があります。そこでは、IPv6が新しいことからIPv4よりセキュアであると信じている人もいるが、それは大きな間違いであると明記されています。プロトコルが新しいということは、むしろ実装上のバグなどが発見され、それがこれから修正されていく必要がある状態であるとも述べられています。さらに、IPv6にとって最大の脅威は、セキュリティを確保しながら運用できる経験の不足であるとしています。

RFC 7381では、IPv6とセキュリティに関する誤解[†2]が3つ紹介されています。1つめの誤解として紹介されているのは、「IPv6のアドレス空間が膨大であることから、/64の空間すべてをスキャンするのが困難である」というものです。これに関しては本章の14.7節で解説します。

2つめは、「IPv6ではIPsecが必須とされているのでIPv4よりもセキュアである」という誤解です。初期のIPv6仕様ではIPsecが必須とされていましたが、現在のIPv6ではIPsecは必須ではありません。IPsecについては、本書では詳しく解説しませんが、14.8節で簡単に紹介しています。

3つめの誤解は、「ブロードキャストがなくなったのでパケットの増幅攻撃が存在しない」というものです。これが誤解であるのは、IPv6でもマルチキャストを転送するときにICMPv6のエラーメッセージや情報メッセージがルータによって生成される可能性があるからです。IPv6でも、IPv4同様に、ICMPv6メッセージに対する制限が必要です。

---

†1 RFC 7381：K. Chittimaneni, T. Chown, L. Howard, V. Kuarsingh, Y. Pouffary, E. Vyncke, “Enterprise IPv6 Deployment Guidelines”, 2014年10月

†2 RFC 7381では、神話、虚構、作り話、通説という意味を持つ「myth」が使われていますが、本書では、それを「誤解」と意訳しました。

14.1.1 IPv6ならではのセキュリティ上の注意点

IPv6ネットワークにも、IPv4ネットワークの場合と同様に、セキュリティ上の注意点が多くあります。WebやSQLといったアプリケーション層に関するセキュリティ、不正なWi-Fiアクセスポイントなどの機器に関連するセキュリティ、さらには、フラッディングやDDoS攻撃、送信元アドレス詐称などについては、IPv4と同様に対策が必要です。

その一方で、IPv6ならではのセキュリティ上の課題や注意点もいくつかあります。

- 近隣探索プロトコルとセキュリティ

  IPv4とIPv6の非常に大きな違いとして、近隣探索プロトコルの存在があります。近隣探索プロトコルに関するさまざまな問題については14.2節でまとめて説明します。また、14.3節では、近隣探索プロトコルに対するセキュリティの仕組みとして考案されたSENDについて紹介します。さらに、14.4節では、不正なRouter Advertisementメッセージに関する問題について解説します。

- プライバシーの問題とTemporary IPv6アドレスの課題

  IPv6アドレスにおけるインターフェース識別子に関しては、ユーザのプライバシー上の問題が指摘されており、それに対応した仕組みとしてTemporary IPv6アドレスが用意されています。ただ、Temporary IPv6アドレスには、ネットワーク管理者がセキュリティ保全のためにログを解析する際に、対象となるIPv6アドレスの機器が追跡できなくなる可能性があります。そのため、Temporary IPv6アドレスを利用する場合には、モニタリングやロギングにおいて、リンク層のアドレスとの関連づけなどの機能が必要になることもあります。さらに、Temporary IPv6アドレスをユーザが利用しないようにする手法として、RFC 7381ではステートフルDHCPv6を利用する手法が紹介されています。これらの問題については14.6節で改めて扱います。

- IPv6拡張ヘッダに伴う管理の複雑化

  IPv6拡張ヘッダの存在により、ルータやファイアウォールにおいて通過させるパケットを識別するためのアクセスコントロールリストの管理が複雑になることがありえます。

- フラグメンテーションに伴うフィルタリングの注意

  IPv6では、IPv4と異なり、パケットのフラグメンテーションがエンドノードにおいてのみ許可されています（第9章参照）。そのため、IPv6では、ICMPv6のPacket Too Bigメッセージを途中経路上でフィルタしないようにする必要があります。

- デュアルスタック環境のセキュリティ

  IPv6とIPv4のデュアルスタック環境では、IPv4のみで構成されるネットワークと比べて、IPv6ネットワークという要素が追加されるため、攻撃者が攻撃対象として狙える要素が倍になります。たとえば、ルータがIPv6とIPv4の両方を有効にしてあるとき、それぞれのプロトコルに対する設定が個別に存在しているため、フィルタの設定もIPv6とIPv4で互いに独立したものです。IPv4に対するフィルタだけを設定していて、IPv6のための設定を忘れている場合もあります。

CC BY NC SA

14.2 近隣探索プロトコルとセキュリティ

IPv6に特有のセキュリティ上の課題として、近隣探索プロトコルに関連するものが挙げられます。近隣探索プロトコルは、IPv6を支える重要な要素であると同時に、IPv4とIPv6の大きな違いでもあります。

近隣探索プロトコルが関連するセキュリティ上の課題については、近隣探索プロトコルが定義されているRFC 4861[†3]のSection 11.1に概要がまとめられています。そこでは、さらに詳細な内容については2004年に発行されたInformationalなRFC 3756[†4]が参照されています。ただし、2004年の段階では14.3節で後述するSEND（SEcure Neighbor Discovery）について記述したRFC 3971も発行されておらず、多少情報が古い部分もあります。

以下、RFC 4861とRFC 3756で紹介されている攻撃のうち、大きな問題となりうるものを紹介します。

14.2.1 近隣不到達性検知へのDoS攻撃

IPv6ノードは、リンク上にあるローカルな宛先への到達性を監視し続けます。一般的な実装では、隣接ノードが到達可能であるかどうかの判断を上位層に依存しています。しかし、上位層でのトラフィックに著しい遅延が発生したり、隣接ノードからの応答が一定時間にわたって確認できない場合には、6.3.5項で説明した近隣不到達性検知（Neighbor Unreachability Detection、NUD）を開始します。具体的には、近隣ノードに対し、Neighbor Solicitationメッセージを送信します。

近隣不到達性検知では、応答がない場合に何度かNeighbor Solicitationメッセージを送信します。何度かリトライが発生したら、近隣キャッシュを破棄し、必要に応じて新たなアドレス解決を行います（6.3.6項参照）。

この近隣不到達性検知に対し、「悪意あるノードから、不到達となるはずのノードに偽装した不正なNeighbor Advertisementメッセージを送信し続ける」というDoS攻撃があります。これにより近隣不到達性検知の正常な実施を妨害し、近隣キャッシュが破棄されない状態が続くという状況が発生します。

14.2.2 DADに対するDoS攻撃

SLAAC（StateLess Address AutoConfiguration）によってIPv6アドレスが自動設定されているホストに対する攻撃として、DAD（Duplicate Address Detection、重複アドレス検出）におけるすべてのメッセージに応答することによってIPv6アドレスが設定されないようにするDoS攻撃が指摘されています。

この攻撃には、DADに利用する2種類のメッセージのいずれを悪用するかによって2つの方法があります。1つは、他のノードによってDADが実施されているかのように偽装するためにNeighbor Solicitationメッセージで応答する方法です。もう1つは、Neighbor Advertisement

---

†3 RFC 4861 : T. Narten, E. Nordmark, W. Simpson, H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, 2007年9月

†4 RFC 3756 : P. Nikander,J. Kempf,E. Nordmark, “IPv6 Neighbor Discovery (ND) Trust Models and Threats”, 2004年5月

CC BY NC SA

メッセージで応答することにより、対象とするIPv6アドレスがすでに存在しているように偽装する方法です。

14.2.3　ルータに関連する攻撃

近隣探索プロトコルには、ルータが関連するメッセージも多くあります。特に、Router Advertisementメッセージに関連しては、非常に大きな影響を与える攻撃があります。Router Advertisementメッセージに関連する攻撃と、それらの攻撃を防ぐための方法については、14.4節で改めて解説します。

Router Advertisementメッセージに関連する攻撃以外にも、RFC 3756では、DoS攻撃によって正規のデフォルトルータを「殺す」攻撃の可能性が説明されています。その状況で、悪意あるノードがデフォルトルータであることを表明すれば、事実上のMan-in-the-Middle攻撃などが可能になります。

Redirectメッセージを利用した攻撃も考えられます。最初の次ホップとなるルータのリンクローカルアドレスを送信元とするRedirectメッセージを偽造することで、そのルータを最初の次ホップとして利用しているホストのトラフィックを不正にリダイレクトすることが可能になるでしょう。

14.2.4　リプレイ攻撃

ここでリプレイ攻撃と呼んでいるのは、ノードが受け取った近隣探索メッセージを攻撃者が横取りし、それを都合がいいタイミングで再送するという攻撃です。メッセージの内容を改ざんするわけではないので、通信の暗号化などでは防げません。

要求に対して応答する場合（Router Solicitationメッセージに対するRouter Advertisementメッセージのような）であれば、応答にNonceを含めなければならないようにすることで、有効なNonceが含まれない古いメッセージを利用したリプレイ攻撃を防げます。

単発で送信されるAdvertisementメッセージやRedirectメッセージをリプレイ攻撃から守る方法としては、タイムスタンプなどがあります。ネットワークに接続されたノード同士の時刻同期の精度が高くなかったとしても、タイムスタンプによって示される時間の差を確認することで、新たに受け取ったメッセージが以前のものよりも新しいことを確認できるでしょう。

14.2.5　遠隔からの攻撃

RFC 3756では、攻撃者が特定のサブネットプレフィックスに対するアドレスを加工し続けるという、近隣探索プロトコルに対する遠隔からの攻撃が説明されています。それらのパケットを受け取ったルータは、サブネット内で該当するアドレスに対応するリンク層アドレスを解決するため、Neighbor Solicitationメッセージをサブネットに対して送信します。実在するサブネットプレフィックスの実在しないインターフェース識別子を問い合わせ続けることになるので、近隣キャッシュがあふれ、事実上のDoS攻撃になります。

このような攻撃は、ネットワーク上で攻撃対象とするデバイスを探すためなどに実施される

cc BY NC SA

アドレススキャンについて解説しているRFC 7707[†5]でも言及されています。RFC 7707では、IPv4では存在しなかったアドレススキャン攻撃の影響として、実際にはネットワーク上に存在しない大量のIPv6アドレスがルータの近隣キャッシュに登録されることで近隣キャッシュがあふれてしまう可能性が言及されています。

遠隔からの攻撃は、ルータに対してだけではなく、ホストに対しても可能です。アプリケーションを騙し、on-linkの宛先に向かってパケットを送り続けさせることで、ホストにおける近隣探索プロトコルの処理を誘発させるというものです。

14.2.6 近隣探索プロトコルとフラグメンテーション

不正な近隣探索メッセージによる影響を軽減するための手段として、14.4.2項で説明するRAガードやIPS（Intrusion Prevention System）などが知られています。しかし、IPv6フラグメンテーションを悪用することで、そういった軽減策をすり抜けることが可能になる場合があります。この問題はRFC 6980[†6]で指摘されています。RFC 6980はStandards Trackであり、近隣探索プロトコルについて定義したRFC 4861と、次節で後述するSENDについて定義したRFC 3971とを更新する内容という位置づけです。

RFC 6980では、すべての近隣探索プロトコルメッセージ（次節で扱うSENDのCertification Path Solicitationメッセージを含む）について、IPv6フラグメントヘッダが含まれるものは無効であるとしています。そのようなパケットを受け取ったノードは、そのパケットを破棄することが求められています。

14.3 SEND（SEcure Neighbor Discovery）

近隣探索プロトコルのもともとの仕様では、近隣探索プロトコルのメッセージはIPsecによって守られる場合があるとされていました。しかし、IPsecの利用にあたっては、鍵交換を自動的に行う手法が必要という制約があります。実際、当初の近隣探索プロトコルにおけるメッセージの送受信に対する暗号化は、手動での鍵管理が可能な状況に限定されていました。

IPsecを利用することなく近隣探索プロトコルに対してセキュリティを提供する仕組みとして考案されたのが、**SEND**（SEcure Neighbor Discovery）です。SENDはRFC 3971[†7]で標準化されています。RFC 3971が発行されたのは2005年のことですが、運用のオーバーヘッドが高く、実装も不十分な状況であることから、SENDの利用は現在でも一般的であるとは言い難い状況です（背景についてはRFC 6980を参照してください）。

14.3.1 SENDのための新しいオプション

SENDでは、近隣探索プロトコルのさまざまな機能を安全にするために、6.5節で紹介した近隣探索メッセージのオプションを新しく4つ利用します。

---

†5 RFC 7707：F. Gont,T. Chown, “Network Reconnaissance in IPv6 Networks”, 2016年3月

†6 RFC 6980：F. Gont, “Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery”, 2013年8月

†7 RFC 3971：J. Arkko, J. Kempf, B. Zill, P. Nikander, “SEcure Neighbor Discovery (SEND)”, 2005年3月

- CGAオプション
- RSA Signatureオプション
- Timestampオプション
- Nonceオプション

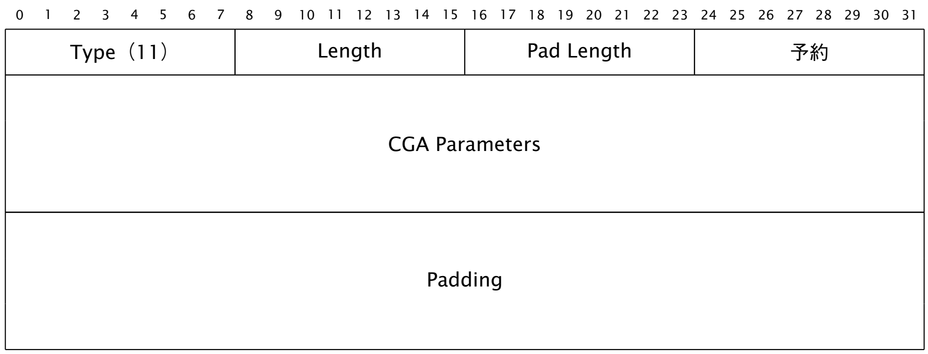
CGAオプションは、各近隣探索メッセージに記載されている内容が、そこに記されたIPv6アドレスの所有者（ownerと呼ばれます）であることを示すためのものです。CGAオプションには、公開鍵とそれに関連するパラメータが含まれます。RSA Signatureオプションは、近隣探索メッセージの内容全体を保護するための署名です。TimestampオプションとNonceオプションは、リプレイ攻撃を防ぐために使います。

以下、それぞれのオプションについて簡単に説明します。

■ CGAオプション

CGAとは、Cryptographically Generated Addressのことで、暗号を使って生成されたアドレスという意味です。CGAには、IPv6アドレスのインターフェース識別子が、公開鍵などのパラメータを利用して得られるハッシュ値と同じになるなどの特徴があります。CGAの具体的な生成方法についてはRFC 3972[†8]を参照してください。

CGAオプションは、送信者のCGAを検証するためにSENDで利用するオプションです。図14.1にCGAオプションのフォーマットを示します。



▶ 図14.1　CGAオプションフォーマット

- **Type（8ビット）**

  近隣探索メッセージのオプションのタイプを指定します。CGAオプションの場合は11です。

- **Length（8ビット）**

  Typeフィールド、Lengthフィールド、Pad Lengthフィールド、予約フィールド、CGA Parametersフィールド、Paddingフィールドをすべて含めたオプション全体の長さを示します。単位は8オクテットです。

---

†8　RFC 3972：T. Aura, “Cryptographically Generated Addresses (CGA)”, 2005年3月

- Pad Length（8 ビット）

  パディングの長さを示すためのフィールドです。単位はオクテットです。

- 予約（8 ビット）

  将来の利用に備えたフィールドです。

- CGA Parameters

  CGA の生成に利用するパラメータを示す可変長フィールドです。

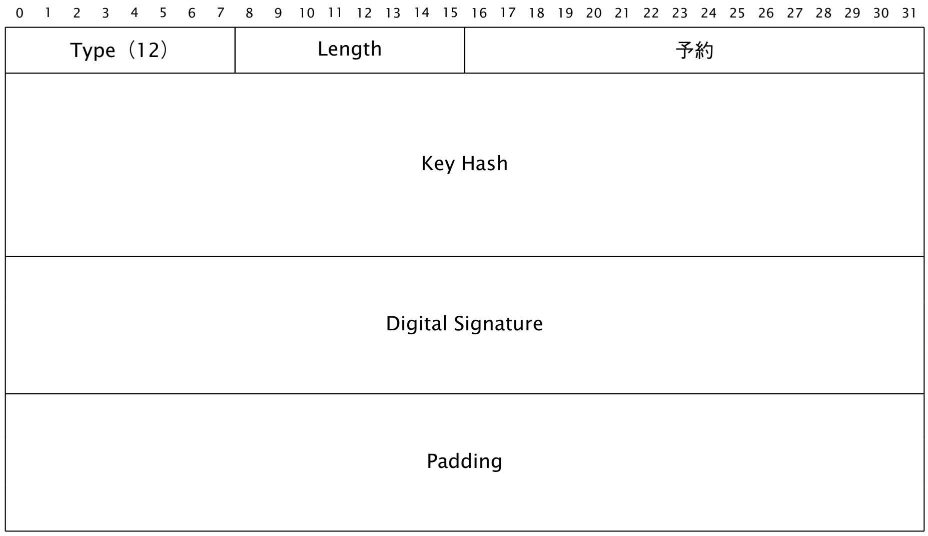
- Padding

  オプション全体の長さが 8 の倍数となるようにするためのパディングです。

■ RSA Signature オプション

SEND では、メッセージ送信者とメッセージに含まれる内容が経路の途中で改ざんされていないことを示すために、公開鍵を利用した電子署名をメッセージに添付できます。そのために使うのが RSA Signature オプションです。署名は、このオプションを含む近隣探索メッセージ全体に対して計算されます。

図 14.2 に RSA Signature オプションのフォーマットを示します。



▶ 図 14.2　RSA Signature オプションフォーマット

- Type（8 ビット）

  近隣探索メッセージのオプションのタイプを指定します。RSA Signature オプションの場合は 12 です。

- Length（8 ビット）

  Type フィールド、Length フィールド、予約フィールド、Key Hash フィールド、Digital Signature フィールド、Padding フィールドをすべて含むオプション全体の長さを示します。単位は 8 オクテットです。

- 予約（16ビット）

  将来の利用に備えたフィールドです。

- Key Hash（128ビット）

  SHA-1ハッシュの上位128ビットを格納するフィールドです。

- Digital Signature（可変長）

  パケットの送信元IPv6アドレス、宛先IPv6アドレス、ICMPv6ヘッダのフィールド、近隣探索プロトコルのメッセージのヘッダ、近隣探索プロトコルのメッセージのオプションなどを署名した可変長データを格納します。

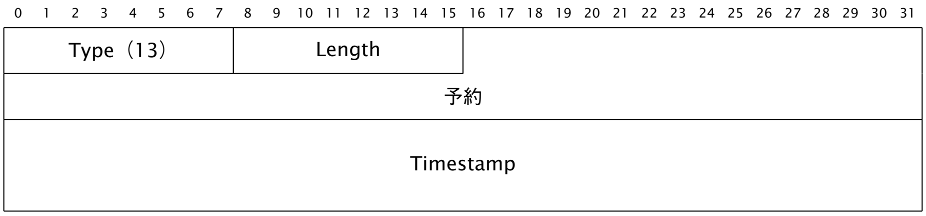
- Padding（可変長）

  オプション全体の長さが8の倍数となるようにするためのパディングです。

■ Timestampオプション

時刻同期が実施されている環境で、リプレイ攻撃を防ぐために使うオプションです。RFC 3971では、NTP（Network Time Protocol）に対する攻撃によって正しくない時刻がSENDノードに設定されてしまう攻撃の可能性が言及されています。このオプションを使ってリプレイ攻撃を防ぐには、SENDノードが正しく時刻を同期できる環境が必要です。

図14.3にTimestampオプションのフォーマットを示します。



▶ 図14.3　Timestampオプションフォーマット

- Type（8ビット）

  近隣探索メッセージのオプションのタイプを指定します。Timestampオプションの場合は13です。

- Length（8ビット）

  Typeフィールド、Lengthフィールド、予約フィールド、Timestampフィールドをすべて含めたオプション全体の長さを示します。単位は8オクテットです。Timestampオプションの場合、このフィールドの値は常に2になります。

- 予約（48ビット）

  将来の利用に備えたフィールドです。

- Timestamp（64ビット）

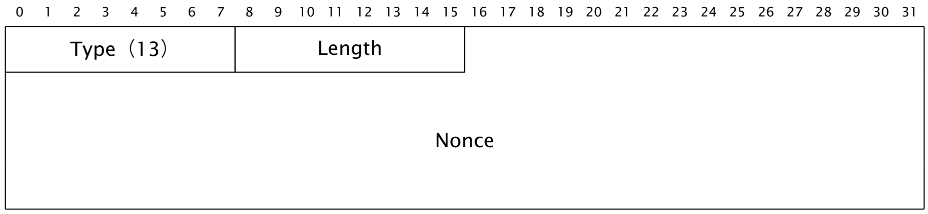
  64ビットの符号なし整数によって示されるタイムスタンプです。1970年1月1日0時

0分（UTC）からの秒数を上位48ビットによって示し、残りの16ビットで1秒の64,000分の1の単位を示します。

■ Nonceオプション

Nonceオプションは、Solicitation系のメッセージへの応答であるようなAdvertisementについて、鮮度を確認するために利用されます。

図14.4にNonceオプションのフォーマットを示します。



▶ 図14.4　Nonceオプションフォーマット

- Type（8ビット）

  近隣探索メッセージのオプションのタイプを指定します。Nonceオプションの場合は14です。

- Length（8ビット）

  Typeフィールド、Lengthフィールド、Nonceフィールドを含むオプション全体の長さを示します。単位は8オクテットです。

- Nonce（可変長）

  Solicitation系のメッセージの送信側が選んだランダムな値（Nonce）を格納します。Nonceは最低でも6オクテットの長さが必要です。Nonceオプション全体の長さが8オクテットの倍数となるようにする必要があります。

NOTE

Nonceオプションは RFC 7527[†9]で定義されている Enhanced DAD でも利用されています。Enhanced DADに関しては7.6.2項を参照してください。

14.3.2 ルータの発見におけるセキュリティ

SENDでは、近隣探索プロトコルにおけるルータの発見（6.2節参照）のセキュリティを向上させるために、トラストアンカーと証明書を利用した認証の方法も定義されています。ホストは、受け取った近隣探索メッセージに対して有効な証明書パスがない場合、認証委任検索（ADD、Authorization Delegation Discovery）と呼ばれる手順を開始します。その際には、

†9 RFC 7527：R. Asati,H. Singh,W. Beebee,C. Pignataro,E. Dart,W. George, “Enhanced Duplicate Address Detection”, 2015年4月

RFC 3971で新たに定義されている2つのICMPv6メッセージと、それらのメッセージに含める2つのオプションを利用します。

- Certification Path Solicitationメッセージ
  ホストからルータに証明書パスを要求するのに利用するICMPv6メッセージです。
- Certification Path Advertisementメッセージ
  ルータがホストからの証明書パスの要求に応答するために利用するICMPv6メッセージです。
- Trust Anchorオプション
  トラストアンカーの名前を指定するためのオプションです。
- Certificationオプション
  Certification Path AdvertisementメッセージでX.509v3証明書を格納するためのオプションです。

これらのメッセージとオプションのフォーマットなどの詳細についてはRFC 3971を参照してください。

14.4 不正なRouter Advertisementメッセージ

近隣探索プロトコルに対する攻撃として、Router Advertisementメッセージに関連するものがあります。攻撃ではなく、運用者の設定ミスなどによって、本来は送信されるべきではないRouter Advertisementが送信されることもあります。そういった不正なRouter Advertisementが問題を発生させる可能性があるのです。

不正なRouter Advertisementメッセージに関しては、以下の3つのInformationalなRFCが発行されています。

- RFC 6104[†10]：正しくないRouter Advertisementメッセージに関する問題をまとめたものです。
- RFC 6105[†11]：RFC 6104で要約された問題への対策として、Router Advertisement Guard（RAガード）という仕組みを提案しています。
- RFC 7113[†12]：RAガードを回避する手段の紹介と、それに対するさらなる防御策を解説しています。

---

†10 RFC 6104：T. Chown, S. Venaas, “Rogue IPv6 Router Advertisement Problem Statement”, 2011年2月
†11 RFC 6105：E. Levy-Abegnoli, G. Van de Velde, C. Popoviciu, J. Mohacsi, “IPv6 Router Advertisement Guard”, 2011年2月
†12 RFC 7113：F. Gont, “Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)”, 2014年2月

14.4.1　問題の概要

RFC 6104では、意図的な攻撃として、偽のRouter Advertisementメッセージが利用される場合があるとしています。とはいえ、意図的な攻撃の場合にはNeighbor Advertisementメッセージなどが利用される傾向があるとして、実際に発生している問題の大半である設定ミスを防ぐための方策について議論されています。

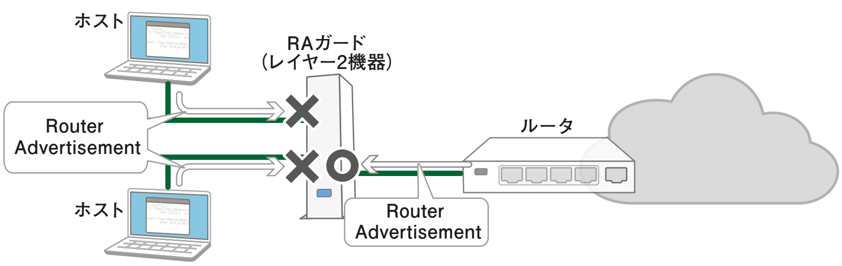
不正な内容になってしまったRouter Advertisementメッセージがリンク内に流れていると、リンクに接続されたホストの一部、もしくは全部について、通信が阻害される可能性があります。たとえば、正しくないRouter Advertisementメッセージに記載されているデフォルトルータを利用しようとすると、通信がまったくできない状態になることもあります。ホストが、正しくないRouter Advertisementメッセージが広告しているネットワークプレフィックスを送信元IPv6アドレスとするパケットを送出してしまうことで、通信できなかったという状況も考えられるでしょう。

RFC 6104では、正しくないRouter Advertisementメッセージが偶発的（accidental）に広告されてしまうシナリオの例として、ネットワーク管理者による誤った設定や、リンクに接続されたホストの管理者が6to4ゲートウェイとしての機能を誤って有効にしてしまうことなどが挙げられています。

RFC 6104では、SENDの利用や、後述するRAガードのようなレイヤー2における対策技術の採用、DHCPv6の利用、ホスト側におけるフィルタの設定など、不正なRouter Advertisementメッセージによる悪影響を軽減（mitigate）する方法もいくつか紹介しています。ただし、どの手法にも一長一短があり、完全な解決方法が記載されているわけではありません。

14.4.2　RAガード

不正なRouter Advertisementメッセージに対する対策として、RFC 6105で**RAガード**（Router Advertisement Guard）が紹介されています。RAガードは、図14.5のように、管理下にあるレイヤー2機器をすべてのIPv6パケットが通過する環境を想定しています。管理下にないレイヤー2機器を通過する通信が発生してしまう状況（シェアードメディア）での利用は想定されていません。図14.5では、ルータからのRouter Advertisementメッセージは通過させる一方で、ホストからのRouter Advertisementメッセージはブロックしています。



▶ 図14.5　RAガードを導入したネットワークの例

RFC 6105では、RAガードを実現する方法として、ステートレスな形態とステートフルな

CC BY NC SA

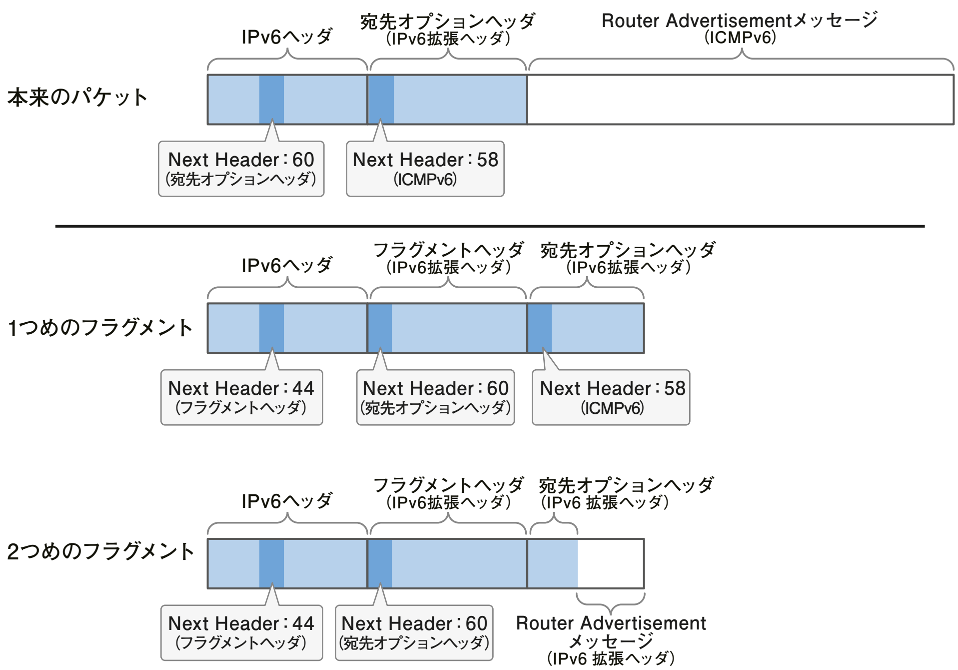
形態の両方を紹介しています。ステートレスな形態としては、たとえば、レイヤー2機器の特定の物理ポートからの入力のみを許可し、それ以外はブロックするという単純なものがあります。Router Advertisementメッセージの送出を許可する送信元IPv6アドレスを設定するといった方法もあります。

ステートフルな形態としては、ある一定期間にRouter Advertisementメッセージの送信を可能とする機器をレイヤー2機器にラーニングさせたうえで、その機器以外からのRouter Advertisementメッセージをブロックするといった方法が紹介されています。

14.4.3　RAガードの回避

RFC 7113では、拡張IPv6ヘッダやIPv6フラグメンテーションを悪用することで、RFC 6105で解説されているRAガードを回避する手法が紹介されています。たとえば、一連の拡張IPv6ヘッダを解析せず、IPv6ヘッダに記載されているNext Headerフィールドのみを解析するようなRAガード実装が存在していることを紹介しています。そのような実装では、何らかの拡張IPv6ヘッダがICMPv6ヘッダの前に置かれているような場合に、RAガードによって遮断されるべきRouter AdvertisementメッセージがRAガードを通過してしまいます。

IPv6フラグメンテーションを悪用することでRAガードを回避する方法としては、Router AdvertisementメッセージのICMPv6ヘッダが最初のフラグメントに含まれないようにすることでRAガードを回避する方法が紹介されています（図14.6）。



▶図14.6　RAガードを回避するようなIPv6フラグメンテーション

図14.6の例では、ICMPv6ヘッダの前に宛先オプションヘッダを挿入したうえで、最初のフラグメントには宛先オプションヘッダのみが含まれるようにし、ICMPv6ヘッダが第2フラグ

CC BY NC SA

メントに含まれるようにしています。このようなフラグメンテーションが行われた場合、フラグメントの再構成を行わない限り、元のパケットがRouter Advertisementメッセージであることがわからないので、RAガードを回避できてしまいます。

実際にRFC 7113で紹介されているRAガード回避テクニックは、最初のフラグメントにすべてのIPv6拡張ヘッダを含めないというものです。しかし、RFC 7112[†13]ではIPv6フラグメンテーションの仕様が更新され、フラグメンテーション時にすべてのIPv6拡張ヘッダが最初のフラグメントに含まれなければならなくなりました（第9章参照）。最新のIPv6基本仕様であるRFC 8200[†14]にはRFC 7112による更新が反映されているので、本書執筆段階では2つめ以降のフラグメントにもIPv6拡張ヘッダを入れるとIPv6の基本仕様に反することになります。そのため、仮にRAガードを回避する目的でそのような状態になったIPv6フラグメンテーションが宛先に届いたとしても、最新のIPv6基本仕様に即した実装であれば破棄されます。とはいえ、古いIPv6仕様のままフラグメンテーションを解釈する機器もしばらくは存在すると思われるので、注意は必要です。

14.5 DHCPv6-Shield

不正なRouter Advertisementメッセージと同様に、不正なDHCPv6メッセージによる攻撃も可能です。たとえば、ステートレスDHCPv6で使われるメッセージを偽装することで、名前解決のためのDNSメッセージを任意のノードに誘導することもできます。DHCPv6メッセージを利用した攻撃を軽減する手法としてRFC 7610[†15]（BCP 119）で提案されているのがDHCPv6-Shieldです。

DHCPv6-ShieldはRAガードによく似ています。レイヤー2でDHCPv6メッセージを送信可能となるポートと、そうでないポートを設定することで、不正なDHCPv6メッセージが送信されることを軽減します。

14.6 IPv6アドレスとプライバシー

SLAACの仕組みでは、IPv6アドレスにおけるインターフェース識別子を生成するのに、IEEE識別子を利用する方法が規定されています。この自動生成されるIPv6アドレスに対しては、プライバシーの懸念があり、RFC 8981[†16]ではSLAACへの拡張としてTemporary IPv6アドレス（匿名IPv6アドレス）が定義されています。ただ、ログ保存やトラブルシューティングに対する悪影響を懸念したネットワークの運用者がTemporary IPv6アドレスを利用できなくしてしまう場合もあるでしょう。

SLAACに限らず、インターフェース識別子の生成に関しては、セキュリティとプライバシーに対する懸念がさまざまに議論されています。そのような議論をInformationalなRFCとして

---

†13 RFC 7112：F. Gont, V. Manral, R. Bonica, “Implications of Oversized IPv6 Header Chains”, 2014年1月

†14 RFC 8200：S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, 2017年7月

†15 RFC 7610：F. Gont,W. Liu,G. Van de Velde, “DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers”, 2015年8月

†16 RFC 8981：F. Gont, S. Krishnan, T. Narten, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6”, 2021年2月

cc BY NC SA

要約したのがRFC 7721[†17]です。

RFC 7721では、IPv6アドレス生成に関連するセキュリティとプライバシーに対する一般的な攻撃手法を下記の4つに分類しています。

- 活動の関連づけ

  IEEE識別子を利用したインターフェース識別子を利用している場合、そのネットワークインターフェースが利用され続ける限り、インターフェース識別子はずっと同一のままです。そのため、IPv6アドレスを利用して、ホストの活動を長期間にわたって関連づけることができます。こうした活動の関連づけでは、IPアドレスだけでなく、DNSにおける名前、HTTPクッキー、ブラウザのフィンガープリントなどの情報も利用されます。NATとDHCPv4が同じホストに対して異なるIPv4アドレスを割り当てるIPv4に比べて、IPv6では、この攻撃の効果が大きくなる可能性が指摘されています。

- 位置トラッキング

  固定的なインターフェース識別子が利用されている場合、そのホストがネットワークを移動したことをインターフェース識別子から判別できます。それにより、そのホストがどのように物理的に移動したかを、ネットワークプレフィックスの位置を追跡することで検知できるようになります。RFC 7721では、この特性を利用した能動的な攻撃手法も紹介されています。同じIPv6リンクに接続したサーバ、もしくは途中経路上にある機器がインターフェース識別子を学習することで、遠隔地にあるネットワークからその機器の存在を確認できるようになります。IPv4では、新しいネットワークに接続したときにまったく新しいIPv4アドレスが割り当てられるので、このような攻撃は可能ではないとされています。

- アドレススキャニング

  アドレススキャニングに関しては、RFC 7707で論じられています。14.7節で改めて説明します。

- デバイスの持つ脆弱性への攻撃

  IEEE識別子は製造者を示す情報を含むので、IEEE識別子がインターフェース識別子として利用される場合には、そのネットワークインターフェースに関するデバイス情報が漏洩しています。その機器固有の脆弱性が存在する場合など、デバイス情報が漏れることで攻撃者に有利な状況になる可能性が考えられます。

IPv6アドレスにおけるインターフェース識別子の生成方法は、IEEE識別子による方法だけでなく、さまざまな手段があります。RFC 7721では、インターフェース識別子の生成方法ごとに、プライバシーやセキュリティの問題が整理されています。

---

†17　RFC 7721：A. Cooper, F. Gont, D. Thaler, “Security and Privacy Considerations for IPv6 Address Generation Mechanisms”, 2016年3月

CC BY NC SA

- **IEEE識別子から自動生成されたアドレス**

  アドレスが利用されている間、活動の関連づけと位置トラッキングが可能です。アドレススキャニングとデバイスの持つ脆弱性への攻撃も可能です。

- **手動で静的に設定されたアドレス**

  アドレスが利用されている間、活動の関連づけと位置トラッキングが可能です。アドレススキャニングとデバイスの持つ脆弱性への攻撃については、静的なアドレスの生成方法によっては可能になるでしょう。

- **固定のTemporary IPv6アドレス**

  ランダムに生成されるTemporary IPv6アドレスを固定的に使う方法で、RFC 7721によればWindowsで利用されています。アドレススキャニングとデバイスの持つ脆弱性への攻撃の可能性はなくなりますが、アドレスが利用されている間は活動の関連づけと位置トラッキングが可能です。

- **CGA**

  SENDで利用されるCGAでは、アドレススキャニングとデバイスの持つ脆弱性への攻撃に加え、位置トラッキングができなくなります。生成に使う公開鍵などの情報が有効な間、活動の関連づけは可能です。

- **RFC 7217の方法**

  RFC 7217[†18]で規定されている、リンクごとにランダムで一意のインターフェース識別子を各ネットワークインターフェースに対して生成する方法です。アドレススキャニング、デバイスの持つ脆弱性への攻撃、位置トラッキングはできません。リンクに接続されている間は活動の関連づけは可能です。

- **Temporary IPv6アドレス**

  Temporary IPv6アドレスのみを利用している機器は、そのアドレスが有効な間だけは活動の関連づけもできますが、アドレススキャニング、デバイスの持つ脆弱性への攻撃、位置トラッキングはできません。

- **DHCPv6**

  アドレスが有効な間は活動の関連づけが可能です。デバイスの持つ脆弱性への攻撃、位置トラッキングはできません。アドレススキャニングについては、DHCPv6サーバの実装によります。

IPv6アドレスの中にはIPv4アドレスが埋め込まれるものもあります。RFC 7721では、その例として、第22章で説明するTeredoやISATAPといったトンネル技術が紹介されています。また、IPv6アドレスの中にIPv4アドレスとポート番号が含まれる仕組みの例として、第24章で説明するLightweight 4over6、MAP-T、MAP-EといったIPv4/IPv6共存技術が紹介されています。RFC 7721では、それらのプロトコルによって使われるIPv6アドレスは、IPv4アドレス

†18 RFC 7217：F. Gont, “A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)”, 2014年4月

から引き継がれたものであるため、プライバシーやセキュリティ上の問題も埋め込まれたIPv4と同様であるとしています。

14.7 IPv6サブネットに対するスキャン

IPv6のアドレス空間は、IPv4のアドレス空間と比べて広大です。一般的に利用されている/64のネットワークプレフィックスでは、そのサブネット内で2の64乗個に近いIPv6アドレスを使えます。これは、IPv4のアドレス空間全体で使える2の32乗個より、さらに2の32乗倍も多い数です。

IPv6の広大なサブネットに対してIPv4と同様の方法でアドレススキャン攻撃を行うのは困難です。「IPv6ネットワークにおけるネットワーク偵察（Network Reconnaissance in IPv6 Networks）」と題され、IPv6におけるアドレススキャン攻撃に関して論じているInformationalなRFC 7707でも、IPv4と比べてサブネットのアドレス空間が膨大なIPv6ではIPv4と同様の方法でアドレススキャンを行うことは困難であるとしています。

ただし、RFC 7707では、アドレススキャンを行うアドレスを絞り込むことが可能であると述べられています。RFC 7707で紹介されている手法としては、SLAACなどを利用してModified EUI-64識別子がIPv6アドレスに利用される場合には上位6ビットめにあるuniversal/localビットが1になること、48ビットのMACアドレスのOUIと残りのビットの間に`0xfffe`が挿入されることなどを利用して、アドレススキャンを行う対象を絞り込めるとしています。さらに、OUIとして利用されるベンダーをある程度まで絞り込める場合もあります。たとえば、スキャン対象がサーバ仮想化技術を使っている場合、VirtualBoxであればOUIが`08:00:27`、VMware ESX Serverであれば OUIが`00:05:69`などとなることが紹介されています。

外部ネットワークからのアドレススキャン攻撃では、Temporary IPv6アドレスが利用されている場合であっても通常のグローバルアドレスは設定されることや、ステートフルDHCPv6や手動設定で`2001:db8::1`のようにインターフェース識別子の上位ビットの多くを0とする設定が多いことなども、スキャン対象のアドレスを絞り込むのに利用できます。ローカルネットワークでのアドレススキャンであれば、全ノードリンクローカルマルチキャストアドレスである`ff02::1`に対してICMPv6 Echoリクエストを送るという手法もあります。

14.8 IPsec

ネットワーク層において認証や暗号化といったセキュリティを実現する仕組みとして、IPsecがあります。IPsecに関連するRFCは数多くありますが、IPsecが実現すること、そのための技術的な要素、IPパケットの具体的な処理など、IPsec対応システムの基本的なアーキテクチャについてはRFC 4301[†19]で示されています。

IPsecを利用するホストやルータでは、ある宛先IPアドレスに対してどのようなセキュリティプロトコルで通信を行うかを保持しておき、それに応じて単方向の接続に対するセキュリティを実現します。IPsecで利用できるセキュリティプロトコルとしては、IPヘッダを含めたパケット全体について送信元の認証ができるAuthentication Header（AH）と、IPヘッダ以外

[†19] RFC 4301：S. Kent, K. Seo, “Security Architecture for the Internet Protocol”, 2005年12月

CC BY NC SA

についてパケットの認証と暗号化を提供するEncapsulating Security Payload（ESP）があります。IPv6では、それぞれに必要な情報を、IPv6拡張ヘッダの「認証ヘッダ」（プロトコル番号51）および「Encapsulating Security Payloadヘッダ」（プロトコル番号50）として、送出する各パケットに付加します。認証ヘッダについてはRFC 4302[†20]、Encapsulating Security PayloadヘッダについてはRFC 4303[†21]で定義されています。

NOTE

IPsecは、IPv6とともに仕様が策定されてきた技術です。しかし、IPv6だけで利用するための仕組みではなく、IPv4とIPv6の両方で利用できます。IPv6拡張ヘッダのような仕組みがないIPv4では、AHとESPにそれぞれ専用のヘッダ（およびトレイラ）を利用します。

かつては、IPv6を実装しているノードに対し、IPsecの実装が必須とされていました。実際、IPv6対応ノードが備えるべき機能をまとめたRFC 4294[†22]では、IPsecに関連するRFC 4301、RFC 4302、RFC 4303のすべてを実装することが必須とされています。しかしRFC 4294は2006年に発行されたものであり、2011年に発行されたRFC 6434[†23]によって上書きされています。RFC 6434は、IPv6対応ノードにおけるIPsecの実装に関してはMUST（必須）ではなくSHOULD（推奨）としており、IPsecを実装しないことも可能です。RFC 6434は、2019年にRFC 8504[†24]によって上書き廃止となっていますが、IPsecの実装がSHOULDであることはRFC 6434と同様です。

もう一点注意が必要なのが、かつてIPsecがMUSTとされていたのは実装であって、IPv6を利用したすべての通信がIPsecで行われなければならないとされていたわけではないことです。RFC 6434によってMUSTがSHOULDへと変更される前であっても、IPsecの適用や運用がIPv6で必須だったわけではないのです。

14.9 ICMPv6を無条件にすべてフィルタリングすべきではない

IPv6では、IPv4では他のプロトコルなどとして実現されていた各種の機能がICMPv6に統合されています。そのため、ICMPv6をファイアウォールなどで完全にフィルタリングしてしまうと、通信障害を発生させる要因になりえます。とはいえ、ICMPv6メッセージに対してまったくフィルタリングを実施しなければ、それによってセキュリティ上のリスクが上昇するという面もあります。そこで、ICMPv6をファイアウォールでフィルタリングする際の注意点が、RFC 4890[†25]としてまとめられています。

RFC 4890では、自らが運用しているサイトの外から送信されてくる以下のようなICMPv6メッセージについてはフィルタすべきではないとしています。

†20 RFC 4302：S. Kent, “IP Authentication Header”, 2005年12月
†21 RFC 4303：S. Kent, “IP Encapsulating Security Payload (ESP)”, 2005年12月
†22 RFC 4294：J. Loughney, “IPv6 Node Requirements”, 2006年4月
†23 RFC 6434：E. Jankiewicz, J. Loughney, T. Narten, “IPv6 Node Requirements”, 2011年12月
†24 RFC 8504：T. Chown, J. Loughney, T. Winters, “IPv6 Node Requirements”, 2019年1月
†25 RFC 4890：E. Davies, J. Mohacsi, “Recommendations for Filtering ICMPv6 Messages in Firewalls”, 2007年5月

CC BY NC SA

- Destination Unreachableメッセージ（タイプ1）のすべてのコード
- Packet Too Bigメッセージ（タイプ2）
- Time Exceededメッセージ（タイプ3）のコード0のみ
- Parameter Problemメッセージ（タイプ4）のコード1とコード2のみ

さらに、RFC 4890では、ICMPv6のエコー要求メッセージ（タイプ128）とエコー応答メッセージ（タイプ129）もフィルタすべきではないとしています。その理由として、ICMPv6のエコー要求メッセージとエコー応答メッセージがTeredo（22.2節参照）プロトコルにおいて重要な役割を果たすこと、IPv4でのICMPネットワークスキャンと比べてICMPv6によるネットワークスキャンのリスクが小さいことを挙げています。

逆に、フィルタすべきICMPv6メッセージとしては、自らが運用しているドメイン外からのNode Information Queryメッセージ（タイプ139）、Node Information Responseメッセージ（タイプ140）、Router Renumberingメッセージ（タイプ138）などが挙げられています。その他、IANAによって内容が規定されていない未知のICMPv6タイプを持つICMPv6メッセージについても、フィルタすべきものとして紹介されています。

14.10　トンネル技術が抱える問題

一般に、トンネル技術にはセキュリティ上のリスクがあります。IPv4/IPv6共存技術にはトンネル技術を採用しているものが多くあり（第22章参照）、それらはトンネル技術そのものが持つセキュリティリスクに晒されることになります。

こうした問題については、RFC 6169[†26]にまとめられています。以下、RFC 6169で紹介されている懸念をいくつか紹介します。

14.10.1　ファイアウォールなどのセキュリティ機器が未対応であることによる問題

ファイアウォールなどのセキュリティ機器が、IPv4/IPv6共存技術に対応していない場合、トンネルの内部を流れるトラフィックを識別できない可能性があります。DPI（Deep Packet Inspection）を行うようなファイアウォールであっても、IPv4/IPv6共存技術で利用されているトンネル技術に対応していなければ、それを利用してセキュリティ機能を迂回できてしまうでしょう。

14.10.2　送信元を偽装する目的に利用可能

トンネル技術を利用して、送信元IPアドレスの偽装などが可能になる場合があります。たとえば、RFC 2827[†27]では、Ingressフィルタを利用して送信元IPアドレス偽装を防いでいるネットワークにおいて、トンネルを利用することでIngressフィルタを回避できる例が紹介されています。IngressフィルタやEgressフィルタは、トンネルが終端されているネットワーク

†26　RFC 6169：S. Krishnan, D. Thaler, J. Hoagland, “Security Concerns with IP Tunneling”, 2011年4月
†27　RFC 2827：P. Ferguson, D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, 2000年5月

CC BY NC SA

に導入する必要があります。

14.10.3 外部から内部への攻撃

ユーザが張ったトンネルを経由することによって、外部から内部への攻撃が可能になる場合もあります。ユーザ自身が気がつかないままIPv4/IPv6共存機能が稼働してトンネルを作成している場合もあるので注意が必要です。

14.10.4 トンネル用のアドレスからプロファイリングが可能

トンネル技術によっては、トンネル用のアドレスからクライアントのプロファイリングが可能です。たとえば、あるトンネル技術のためのプロトコルを利用しているという事実から、そのトンネル技術のためのプロトコルが実装されているプラットフォームを利用していることが推測できるでしょう。初期状態でそのプロトコルが利用できる状態になっているプラットフォームが限定されている場合には、さらにプラットフォームの特定が可能になります。たとえば、Teredoアドレスを利用しているクライアントであれば、まずWindowsであると推測可能です。

14.10.5 悪意あるトンネルサーバ設定の変更による盗聴

トンネル技術のクライアント側設定で接続先サーバを変更することにより、Man-in-the-Middle攻撃が可能になる場合があります。サーバ認証なしでトンネルが利用される状況では、ユーザが攻撃に気がつきにくくなる可能性もあります。DNSサーバに対するキャッシュポイズニングと組み合わせた攻撃もありえます。

14.11 IPv4-Mapped IPv6アドレスの問題

IPv6にはIPv4-Mapped IPv6アドレスという仕組みがあり、IPv6を利用してIPv4アドレスと通信しているかのような挙動を実現できます（3.9節参照）。便利な仕組みですが、IPv6を通じてIPv4の通信を偽装することで、ファイアウォールなどのセキュリティ機器によるフィルタをすり抜けてしまうといったセキュリティ上の懸念があります。そのため、IPv4-Mapped IPv6アドレスを使うつもりがないアプリケーションをIPv6ソケットを利用して開発する場合には、IPv6ソケットに対して明示的に`IPV6_V6ONLY`オプションを設定しておくことをお勧めします。

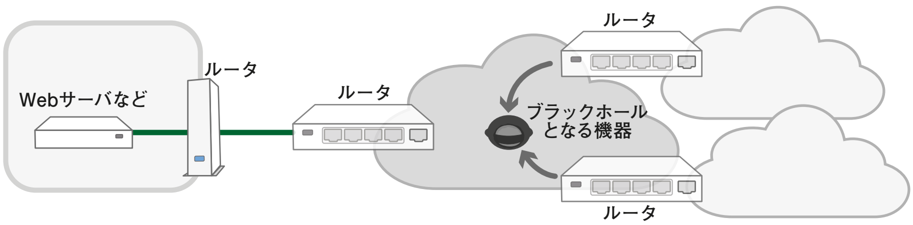
14.12 ブラックホール用IPv6アドレス

RFC 3882[†28]やRFC 5635[†29]では、経路を制御することで攻撃パケットを破棄できるようにする、ブラックホール技術が紹介されています。図14.7のように、攻撃されている対象に対する宛先経路（次ホップルータ）を別の場所へと向けてしまうことで、攻撃トラフィックがブ

†28 RFC 3882：D. Turk, “Configuring BGP to Block Denial-of-Service Attacks”, 2004年9月

†29 RFC 5635：W. Kumari, D. McPherson, “Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)”, 2009年8月

ラックホールへ吸い込まれるように破棄されます。次ホップルータとして監視機器などを指定することで、攻撃トラフィックを監視することも可能です。



▶ 図14.7　ブラックホール技術

ブラックホール技術には、ルータにあまり多くの負荷をかけずに実施できるという特徴があります。ルータにとっては、宛先経路がブラックホールに向かっているだけで、パケット転送処理としては通常と同じだからです。一般的なルータは、受け取ったパケットをルーティングテーブルどおりに転送することに最適化されているので、それ以外のことをやろうとするとCPUやメモリなどの資源を消費してしまいます。ルータでの計算機資源の浪費は通信品質の低下を招くので、ルータにかかる負荷は最小限に抑える必要があります。

ブラックホール技術を利用するときは、次ホップとしてどのようなIPアドレスを使うのかを運用者が選択する必要があります。IPv4では、プライベートIPv4アドレスや、ドキュメント用の`192.0.2.0/24`が使われることがあります。IPv6では、このような用途のための専用のIPv6アドレスが`0100::/64`としてRFC 6666[†30]で定義されています。

[†30] RFC 6666：N. Hilliard, D. Freedman, “A Discard Prefix for IPv6”, 2012年8月

CC BY NC SA

第15章

プログラマにとってのIPv6対応

IPv6対応は、ネットワーク層だけの話ではありません。IPv4とIPv6は異なるプロトコルなので、アプリケーション層におけるIPv6対応が必要な場合もあります。ネットワーク上の機器のIPv6対応や、ホストのOSにおけるIPv6対応が完了し、それらでIPv6機能が有効になっていても、個別のアプリケーションがIPv6対応していなければアプリケーションを使う人はIPv6を利用できないのです。

本書執筆時点では、ネットワーク層の技術に直接関連する職業以外で、アプリケーションプログラミングにおけるIPv6対応はあまり考慮されていないことが多いと予想されます。「IPv6対応に興味はあるけれど、すぐには着手できないので先でよい」という判断をしているプログラマも少なくないでしょう。アプリケーション開発の現場でIPv6対応プログラミングのノウハウが蓄積されるまでには、もう少し時間がかかりそうです。

とはいえ、これまでIPv4のみを考慮することが前提であったネットワークアプリケーションのプログラミングでも、今後は徐々にIPv6対応が必須となっていくでしょう。本章では、アプリケーションプログラムにおけるIPv6対応のヒントになるような情報をまとめます。

15.1 Socket APIとIPv6

実のところ、高度なAPIを有するライブラリが増えていることもあり、開発者がいっさい気にしなくてもアプリケーションのIPv6対応ができる環境が整いつつあります。とはいえ、それらのアプリケーションの内側で何が起きているかを知ることも重要です。場合によっては、自分でSocket APIを利用するIPv6対応プログラムを書かざるを得ないこともあるでしょう。

この節では、Socket APIにおけるIPv6対応について解説します。Socket APIそのものはPOSIXで規定されており、システムごとに多少の違いはあるものの、多くのシステムで同じような記述が可能です。

cc BY NC SA

15.1.1 IPv6のSocket APIに関するRFC

IPv6におけるSocket APIの基本は、RFC 3493[†1]で定められています。RFC 3493は、BSDにおけるSocket APIの実装をIPv6化するための情報を定めたものです。IPv4を前提とするSocket APIとの相違、IPv6アドレスファミリ（`AF_INET6`）およびIPv6プロトコルファミリ（`PF_INET6`）の追加、IPv6アドレスに対応した構造体（`sockaddr_in6`、`sockaddr_storage`）、ネットワークインターフェースに関するAPI、ソケットオプションなどの基本的な事項が記載されています。

IPv6拡張ヘッダやICMPv6に関連するAPIなどは、RFC 3542[†2]に記載されています。

マルチキャストについては、送信元に関係なく特定のマルチキャストグループ宛のパケットを受け取るためのソケットオプションがRFC 3493で扱われています。特定の送信元アドレスからのマルチキャストのみを対象としてグループに参加できるような、いわゆるSSM（Source Specific Multicast）と、SSMに対応したIPv6のマルチキャストプロトコルであるMLDv2については、RFC 3678[†3]で関係するSocket APIが定義されています。

さらに、RFC 5014[†4]では、デフォルトIPv6アドレス選択（13.1節参照）における送信元アドレスの選択ルールに関して、アプリケーションから選択ルールを制御するためのAPIが提案されています。デフォルトIPv6アドレス選択における送信元アドレスの選択を上書きすることになるので、結果として`getaddrinfo()`で宛先アドレスを取得する際の候補の順番にも影響をもたらします。

RFC 3493、RFC 3542、RFC 3678、RFC 5014は、いずれも参考情報という位置づけのInformationalなRFCです。標準という位置づけのStandardではないので、Socket APIの実装がこれらのRFCと同じ方式であることが求められているわけではありません。RFC 3493では、Socket APIの標準はISO/IEC、IEEE、The Open Groupが合同で開発しているPOSIXであり、将来の標準化については“Open Group Base Working Group”を参照するように説明しています[†5]。

NOTE

APIを提案しているRFCではありませんが、IPv4とIPv6が共存する環境におけるIPv6アプリケーション開発の外観を示しているRFC 4038[†6]では、Socket APIを利用したIPv6対応プログラムのサンプルが記載されています。

---

†1 RFC 3493：R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens, “Basic Socket Interface Extensions for IPv6”, 2003年2月

†2 RFC 3542：W. Stevens, M. Thomas, E. Nordmark, T. Jinmei, “Advanced Sockets Application Program Interface (API) for IPv6”, 2003年5月

†3 RFC 3678：D. Thaler, B. Fenner, B. Quinn, “Socket Interface Extensions for Multicast Source Filters”, 2004年1月

†4 RFC 5014：E. Nordmark, S. Chakrabarti, J. Laganier, “IPv6 Socket API for Source Address Selection”, 2007年9月

†5 2021年11月時点では`https://pubs.opengroup.org/onlinepubs/9699919799/`で参照できます。

†6 RFC 4038：M-K. Shin, Y-G. Hong, J. Hagino, P. Savola, E. M. Castro, “Application Aspects of IPv6 Transition”, 2005年3月

CC BY NC SA

15.1.2 IPv6のみで通信するアプリケーションの動作

Socket APIを使って、IPv6のみで通信するアプリケーションをプログラムする場合には、以下の手順で動作するようにプログラムを書きます。

1. IPv6であることを示すAF_INET6（もしくはPF_INET6）でソケットを作成する
2. 作成したソケットに対し、必要に応じて初期設定を行う
3. 通信相手のIPv6アドレスがなく、FQDN情報しかない場合には、名前解決をしてIPv6アドレス情報を取得する
4. 名前解決の結果が複数ある場合には、最終的にどのIPv6アドレスを利用するのかを判断する必要がある。得られた結果順に接続を試み、最初に成功した結果を利用するという手法を採用してもよい
5. 必要に応じて通信相手との初期接続を確立する（たとえばTCPの場合）
6. ソケットに対して、書き込みや読み込みなどの操作を行う
7. 通信が終了したら、必要がなくなったソケットを閉じる

15.1.3 IPv4もしくはIPv6で通信するアプリケーションの動作

Socket APIを使って、IPv4もしくはIPv6で通信するアプリケーションを作成するには、IPv4およびIPv6の両方のソケットを作成し、以下のような手順で動作するようにプログラムを書きます。

- IPv4ソケット（AF_INET）とIPv6ソケット（AF_INET6）の両方を作成する
- 作成した両方のソケットに対し、必要に応じて初期設定を行う
- 通信相手のIPアドレスがなく、FQDN情報しかない場合には、名前解決をする。このとき、名前解決によって得られたIPv4アドレスはIPv4ソケットに対して利用し、IPv6アドレスはIPv6ソケットに対して利用する
- 必要に応じて通信相手との初期接続を確立する（TCPの場合）。このとき、IPv4を利用するのか、それともIPv6を利用するのかを判断するためのプログラムを書かなければならない。IPv4のシングルスタックもしくはIPv6のシングルスタックの場合は、名前解決の結果が複数であった場合と同じようにすればよい。デュアルスタックの場合には、利用するソケットの種類がIPv4とIPv6で異なるという点に注意が必要になる
- 最終的に利用するソケットに対して、書き込みや読み込みなどの操作を行う
- 通信が終了したら、必要がなくなったソケットを閉じる

上記の通信手順からわかるように、IPv4とIPv6のデュアルスタック環境では、IPv4を使うのか、それともIPv6を使うのかを、アプリケーションが自ら判断して通信する必要があります。広く採用されているのは、IPv6を最初に試して成功すればIPv6で通信を行い、IPv6での通信が失敗した場合にはIPv4を利用する、というものです。しかし、より品質が高いほうを採用すべきという発想の手法もあります（15.2節参照）。

cc BY NC SA

15.1.4 IPv6とIPv4とで利用する関数が違う処理

多くのプログラミング参考書では、IPv4用の関数を利用して通信プログラムを解説しています。しかし、IPv6のSocket APIでは、IPv4向けに用意されているのとは異なる関数を利用する場合があります。これらのAPIについてもRFC 3493で解説されています。ここでは、そのうちのいくつかを紹介します。

- 名前解決にはgetaddrinfo()を使う
  IPv4のみのアプリケーションを書く従来のプログラムでは、名前解決のためにgethostbyname()という関数を利用するのが一般的でした。しかし、gethostbyname()はIPv4の名前解決しか行えず、IPv6は扱えません。IPv6に対応したアプリケーションを書くには、gethostbyname()ではなくgetaddrinfo()を利用します。
- IPアドレスと文字列の相互変換にはinet_pton()、inet_ntopを使う
  IPv4のみのアプリケーションを書く従来のプログラムでは、文字列で書かれたIPv4アドレスを32ビットのバイナリデータに変換するためにはinet_addr()関数、32ビットのバイナリデータから文字列データを得るためにはinet_ntoa()関数がそれぞれ利用されてきました。これらは、IPv6アドレスに対しては使えません。文字列で書かれたIPv6アドレスを128ビットのバイナリデータに変換するためにはinet_pton()を、128ビットのバイナリデータから文字列データを得るためにはinet_ntop()をそれぞれ利用します。

NOTE

RFC 4038の6.2.3. "Binary/Presentation Format Conversion" では、inet_pton()とinet_ntop()はプロトコル依存のコードになるので、flagsにNI_NUMERICHOSTが含まれたgetnameinfo()や、flagsにAI_NUMERICHOSTが含まれたgetaddrinfo()を利用するほうが良いとしています。
さらにRFC 4038のAppendix Aでは、inet_ntop()とinet_pton()は、たとえばリンクローカルアドレスなどでスコープ識別子が利用される場合に、スコープ識別子に関する情報が失われてしまうという問題があることにも言及しています。getnameinfo()とgetaddrinfo()であれば、スコープ識別子が持つ情報が失われません。

IPv6対応を求めるRFC

すべてのネットワーク機器にIPv6対応を求めるRFCが2011年に発行されています。RFC 6540[†7]です。RFC 6540は、主に機器の開発者に向けた内容で、従来はIPv4のみを想定していた「IP対応」という表現について、これからはIPv6対応を含めたものを指して使用すべきであると主張されています。とはいえ、一度普及したハードウェアを置き換えるのは困難なので、これからの「IP対応」機器はIPv6機能も含むべきであると述べられています。

注意が必要なのが、RFC 6540はStandard Trackではなく、BCP（Best Current Practice）である点です。BCPは、「RFC発行時点での最良の手法」という意味を持つ文書な

CC BY NC SA

ので、参考にすればよいといった程度であるとも解釈できます。RFC 6540の執筆者が狙ったのは、顧客側からRFC準拠を根拠としてベンダーらにIPv6対応を促すための布石を打つ効果なのかもしれません。

15.2 単なるIPv6対応では不十分な場合

多くの資料では、「IPv6対応クライアントプログラミング」として、単純に`getaddrinfo()`を利用することを推奨しています。確かに、`gethostbyname()`を`getaddrinfo()`にするだけで、IPv6対応そのものは可能です。

とはいえ、IPv6を優先するというポリシーで書かれたプログラムでは不十分な場合もあります。現状ではまだ、IPv4によって運用されているインターネットのほうが安定している状況も少なくありません。IPv4/IPv6共存技術の設定ミスなどで不具合が発生する場合もあります。ネットワークとして見るとIPv4のほうが有利な状況下で、クライアントがIPv6を常に優先してしまうと、IPv4へのフォールバックに成功するまで通信開始を待つことになる可能性があります。IPv6による接続そのものは可能で、IPv4へのフォールバックが起こらないけれど、IPv6の通信品質のほうが悪い場合には、通信のスループットが十分に出ないという問題が発生してしまいます。

動くものを作るという立場からすれば、「IPv6が使えればIPv6を使う」という判断でも問題はないでしょう。しかし、エンドユーザの視点から見れば、「状況に応じて快適に接続されるほう」を使ってほしいわけです。アプリケーションが動作しうる環境で、快適な動作を実現するには、IPv6とIPv4のうち応答が速いほうや通信品質が良いほうを採用するなど、手の込んだ処理が必要になります。IPv6対応アプリケーションのプログラミングでは、プログラムの動作として問題がないようにエラーハンドリングするだけでなく、実際の通信品質まで考慮した動作がユーザビリティという面で求められるのです。

IPv4とIPv6のどちらを使うべきかという問題意識は1994年から

IPv4を使うべきか、それともIPv4以外を使うべきかに関する問題意識は、IPv6が世に出るよりも前の1994年に発行されたRFC 1671[†8]にまで遡ります。そこでは、IPv4に代わる新プロトコル（つまりIPv6）の立ち上がり時には、設定ミスなどによる**ブラックホール**が存在している可能性があるとしています。ここでブラックホールというのは、DNSではIPv6アドレスが返されるのに、途中のルータがIPv6に未対応であるといった何らかの理由で、実際にはそのIPv6アドレスでは目的のホストに到達できない状況です。IPv4アドレスとIPv6アドレスの両方がホストに登録されていると判明していても、ブラックホールが存在しないとは言い切れません。

RFC 1671では、手動設定ではない方法によって、このようなブラックホール問題に対する解決策を確立することが必須であると記述されています。しかし、残念ながら、この問題に対する根本的な解決策がないまま現在に至っています。

†7 RFC 6540 : W. George, C. Donley, C. Liljenstolpe, L. Howard, “IPv6 Support Required for All IP-Capable Nodes”, 2012年4月

CC BY NC SA

15.3 Happy Eyeballs

インターネットを利用するプロトコルやアプリケーションの多くでは、「名前」を利用して通信相手を指定します。1つの名前に対し、複数のIPアドレスが関連づけられていることもありますが、どのIPアドレスを利用して通信するかは各アプリケーションで決定することになります。名前からIPv4とIPv6の両方のIPアドレスが得られても、そのうち片方ではまったく通信できなかったり、何らかの要因によって十分な通信性能が得られないこともありえます。

そこで、IPv6とIPv4の両方を同時に試しつつ、実際の通信で利用するためのIPアドレスを決定するための手法として、Happy Eyeballsというアルゴリズムが提案されています。本書執筆時点におけるHappy Eyeballsの最新バージョンは、RFC 8305[†9]で規定されているVersion 2です。

Happy Eyeballsの目的は、短時間で接続確立を諦めて次を試せることや、毎回必ず同時に複数の接続開始を行うことによるネットワークへの過度の負荷を避けることです。したがって、Happy Eyeballsは、素早くかつ効率良く通信ができると思われる方法を推測するための仕組みだといえます。とはいえ、IPv6利用を促進するという目的もあり、そのため、IPv6をIPv4よりも優先的に採用することを前提としています。

NOTE

RFC 8305では、NAT64とDNS64を利用してIPv4アドレスが直接指定された場合や、AAAAレコードが示す宛先への到達性に問題がある場合に注意すべき点も紹介されています。また、Path MTUに問題がある場合への対処を別途考慮する必要があることや、Happy Eyeballsアルゴリズムを使うことでIPv4もしくはIPv6での通信に問題が発生する環境を発見しにくくなってしまう可能性などに関しても言及されています。

15.3.1 Happy Eyeballs Version 2の概要

Happy Eyeballs Version 2の動作は以下のような流れになっています。

1. 非同期にDNSクエリを実行
2. 得られた名前解決の結果をソート
3. 非同期に接続確立を試行
4. 得られた接続を採用し、他の接続確立の試行をすべてキャンセル

■ 非同期のDNSクエリを実行

Happy Eyeballsでは、IPv6とIPv4のデュアルスタック環境（第III部参照）におけるDNSの利用を前提としています。そこで、名前解決のためにDNSサーバへ問い合わせを実行する際は、IPv4用のAレコードに対するものと、IPv6用のAAAAレコードに対するものを個別に両方とも行う必要があります。Happy Eyeballs Version 2では、両方のクエリをほぼ同時に実行すべきであるとしています。ただし、IPv6用のAAAAレコードに対する問い合わせを先に行うこ

†8 RFC 1671：B. Carpenter, “IPng White Paper on Transition and Other Considerations”, 1994年8月
†9 RFC 8305：D. Schinazi, T. Pauly, “Happy Eyeballs Version 2: Better Connectivity Using Concurrency”, 2017年12月

cc BY NC SA

とを推奨しています。

注意が必要なのは、IPv6とIPv4の両方のアドレスファミリに対する問い合わせが完了するのを待ち、両方の回答が得られてから接続確立を試みる、という手法が非推奨となっている点です。そのような方法が非推奨となっているのは、片方のアドレスファミリでの回答を受け取りつつ、もう片方の回答に時間がかかってまうような場合に、すでに名前解決の結果が得られているアドレスファミリで接続を確立するまでの時間が長くなってしまうからです。非同期のDNS APIが提供されていないプラットフォームについては、アドレスファミリごとにスレッドを生成し、それぞれのスレッドで同期APIによる名前解決を行うことで、疑似的に非同期の名前解決を実現可能です。

先に触れたように、Happy Eyeballs Version 2ではAAAAレコードを優先的に採用するアルゴリズムを推奨しています。非同期のDNSクエリにより、AAAAレコードの結果が先に到達した場合には、即座にそのAAAAレコードが示すIPv6アドレスに対する接続確立を試行することが推奨されています。一方で、AAAAレコードに対する返答よりも先にAレコードに対する返答が到着した場合には、IPv6を優先するための猶予時間を設定することが推奨されています。RFC 8305では、猶予時間の例として50ミリ秒が推奨されています。その猶予時間中にAAAAレコードに対するネガティブな結果（no errorでno answer、RCode=NoErrorでANCOUNT=0）が返ってきた場合には、すでに受け取っているIPv4アドレスを利用して、接続確立のためのIPアドレスリストのソートや接続確立を試行する段階へと即座に進むべきであるとしています。

複数のキャッシュDNSサーバが存在するとき、名前解決を行うクライアントは、キャッシュDNSサーバとの通信にIPv4を使うのか、それともIPv6を使うのかを選べます（17.1.1項参照）。Happy Eyeballs Version 2では、そのような場合にはDNSクエリのトランスポートとしてIPv6を利用することを推奨しています。

■ 得られた名前解決の結果をソート

名前解決でIPアドレスが得られたら、接続を試行する前に、接続確立の順番を決めるためにIPアドレスのリストを作ることが推奨されています。そのリストに記載されたIPアドレスについて順番に接続確立を試行します。RFC 8305では、このリストに、最初に得られたIPv4アドレスとIPv6アドレスそれぞれ1つだけでなく、得られた結果すべてが含まれていることが重要であるとしています。また、異なるアドレスファミリが交互にリストに登録されることを推奨しています。たとえば、IPv6アドレスが最初の項目であれば、次はIPv4アドレスになるようにします。

このIPアドレスのリストを優先順にソートします。その際、13.1.2項で説明したデフォルトIPv6アドレス選択（RFC 6724[†10]）における宛先IPv6アドレスの決定ルールに基づいてソートを行う必要があります。過去の接続成功時のRTTのような情報を参考にすることも可能ですが、判断基準として優先度をあまり高くしないことが推奨されています。送信元IPv6アドレスの決定ルールに関しては、リストにある宛先IPv6アドレスごとに考慮されるべきとされて

[†10] RFC 6724：D. Thaler, R. Draves, A. Matsumoto, T. Chown, “Default Address Selection for Internet Protocol Version 6 (IPv6)”, 2012年9月

CC BY NC SA

いますが、Happy Eyeballs Version 2の範疇外であるとも明記されています。

■ 非同期に接続確立を試行

その時点における優先順が反映されたリストが構築された段階で、クライアントは接続確立を試みます。RFC 8305では、ネットワークへの負荷を過度に上昇させないために、リストにあるIPアドレスに対して複数の接続確立を同時に開始しないことを推奨しています。そうではなく、1つの接続確立を試行して一定時間でその試行が成功しなかった場合に、次の試行を開始するようにします。接続確立を試行してから、次の試行まで成功を待つ時間としては、250ミリ秒が推奨されています。

接続確立の試行が成功した時点でその接続を採用し、他の接続試行中のセッションをすべてキャンセルすることが推奨されています。

NOTE

すでに得られた名前解決の結果がDNS Push Notification[†11]などによって変更された場合には、その内容をリストに反映されることが望ましいとされています。ただし、すでに接続確立を試行しているIPアドレスに対して変更が発生した場合には、その接続確立の試行を中止すべきではありません。

15.3.2 Happy Eyeballs Version 1と2の違い

RFC 8305で規定されているHappy Eyeballs Version 2は、RFC 6555[†12]に記述されていたアルゴリズム（便宜的にHappy Eyeballs Version 1と呼びます）を廃止するものです。Happy Eyeballs Version 1も、Version 2同様に、Standard TrackのRFCでした。

RFC 6555は、RFC 8305と比べると、アルゴリズムそのものを規定するというよりも、アルゴリズムに要求される事項をまとめたものになっています。RFC 8305では、より具体的なアルゴリズムが示されているといってよいでしょう。RFC 8305で示されているVersion 2のアルゴリズムの実装は、RFC 6555の要求にも従うものでありながら、さらに品質を向上させたものであるとされています。

Version 1からVersion 2になる際に追加された点は以下のように要約できます。

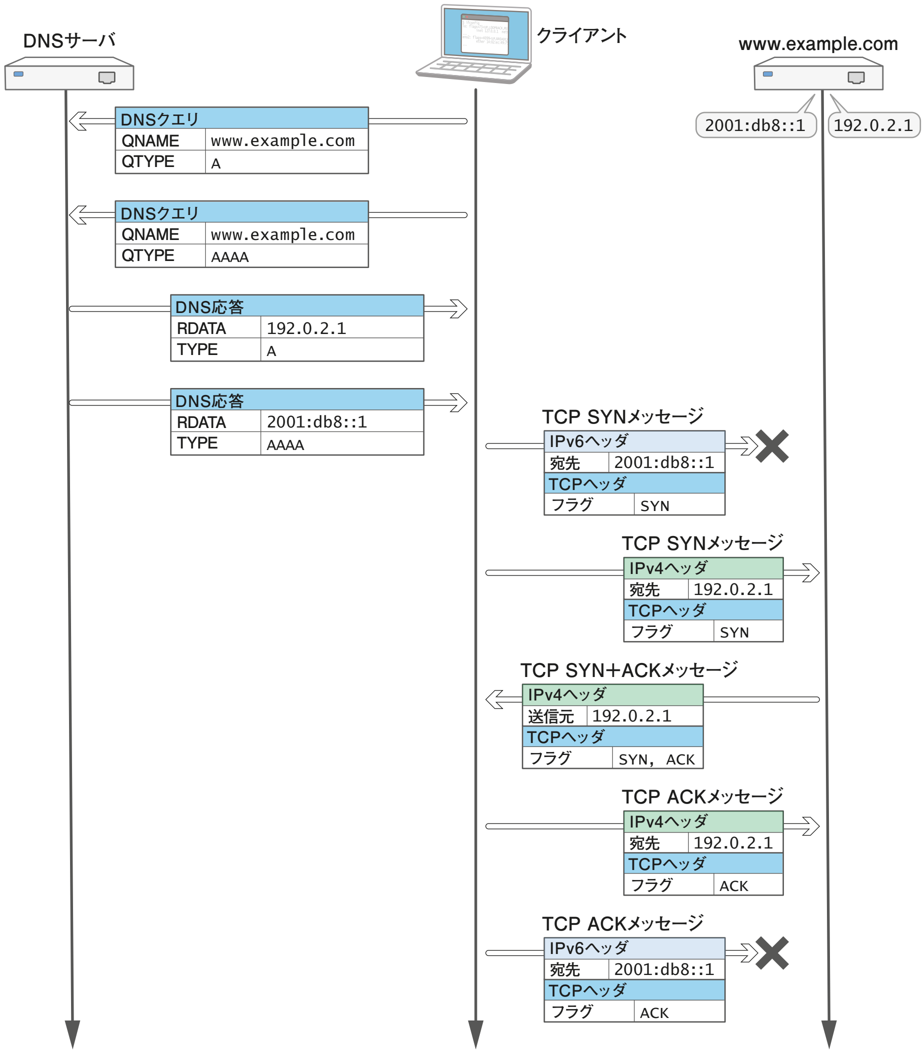
- IPアドレスを得るためのDNSクエリの方法
- 各アドレスファミリごとに複数のIPアドレスがある場合の処理
- 接続確立の試行中にDNS Push NotificationやTTL切れによりDNSに関連する情報が更新された場合の処理
- 過去の通信履歴を利用する方法
- NAT64とDNS64によるIPv6-onlyネットワークをサポートする方法

1つめの「IPアドレスを得るためのDNSクエリの方法」について補足しておきましょう。

†11　Tim Wicinski, “DNS Push Notifications”, Internet-Draft, `https://datatracker.ietf.org/doc/draft-ietf-dnssd-push/`

†12　RFC 6555：D. Wing, A. Yourtchenko, “Happy Eyeballs: Success with Dual-Stack Hosts”, 2012年4月

CC BY NC SA



▶ 図 15.1 RFC 6555 Happy Eyeballs の流れ

Version 2 では、IP アドレスを得るための DNS サーバに対するクエリに関して、非同期 API の利用が推奨されています。この部分について、Version 1 では、`getaddrinfo()` というブロッキングする同期 API の利用が前提となっていました。Version 2 で `getaddrinfo()` のような同期 API を使う場合には、複数のスレッドを利用して疑似的に非同期の DNS クエリを実施することが推奨されています。

Version 1 では、DNS を利用した名前解決の結果が出たあとで、どのように TCP 接続を実施するかにも重点が置かれていました。RFC 6555 では、Happy Eyeballs の流れとして図 15.1 のような例が紹介されています。

この図では、IPv4 と IPv6 の両方で同時に TCP SYN が送信されますが、IPv6 による TCP SYN パケットは喪失しています。IPv4 による TCP 接続は確立しますが、アプリケーションが設定

CC BY NC SA

したタイムアウトまで待ち、そのタイムアウトになるまではIPv6によるTCP接続の試行を続けます。複数のTCP接続を並行して確立しようとしつつ、最初に成功したTCP接続を利用することで、IPv4とIPv6のどちらを利用するかを自動的に判断するわけです。

参考までに、RFC 6555に掲載されているGoogle ChromeとMozilla Firefoxで採用されているHappy Eyeballs version 1のアルゴリズムを紹介します。

1. `getaddrinfo()`を利用して名前解決を行う
2. `getaddrinfo()`から受け取ったIPアドレスのリストの順番に接続を試みる
3. 接続の試みが短時間で確立しない場合（FirefoxやChromeでは300ミリ秒）、異なるアドレスファミリの最初のIPアドレスでの接続を試みる。直前に失敗したアドレスファミリがIPv6であれば、IPv4を試すことになる
4. 最初に確立した接続を利用する

なお、Happy Eyeballs version 1の実装については、RFC 6556[†13]で実験結果が報告されています。

NOTE

Happy Eyeballs的な手法には、RFC 6555やRFC 8035のアルゴリズムだけでなく、さまざまなものが提案されています。たとえば、2015年の中旬にAppleがOS X El Capitanで実装した手法があります。Appleの手法では、IPv4とIPv6の両方に対する名前解決を実施しつつ、もし最初に届いた結果がIPv4の場合には25ミリ秒待ってからTCP接続の確立を試みることで、IPv6を優先的に使うための工夫をしています[†14]。

15.4　IPv6ソケットとIPv4-Mapped IPv6アドレス

IPv4のソケットはSocket APIを`AF_INET`というオプションで使うことにより生成し、IPv6のソケットは`AF_INET6`というオプションを使うことで生成します。このようにIPv4とIPv6のソケットは違うものですが、IPv4-Mapped IPv6アドレスを利用することで、IPv6ソケットを通じてIPv4の通信ができます。IPv4-Mapped IPv6アドレスを利用できるような機能の多くでは、デフォルトでこの機能が有効になっています。この機能を無効にするには、`IPV6_V6ONLY`というオプションを付けて`setsockopt`を実行します。

IPv4での通信ができないようにIPv6ソケットのみを使ってプログラムを書き、意図的にIPv4ソケットを生成していない場合でも、IPv4-Mapped IPv6アドレスの機能を無効にしていないと、気がつかずにIPv4で通信ができる状態になっていることがあるので注意が必要です。たとえば、`http://[::ffff:192.0.2.1]/`といったURLを使って接続が可能なクライアントプログラムもあります。

逆に、IPv4だけで使っているつもりでも、IPv6対応のためにIPv6ソケットでサーバが実装されていれば、知らずにIPv6側でもサーバが稼働していることもあります。たとえば、IPv4

†13 RFC 6556：F. Baker, “Testing Eyeball Happiness”, 2012年4月
†14 David Schinazi, “[v6ops] Apple and IPv6 - Happy Eyeballs”, 2015年7月, https://www.ietf.org/mail-archive/web/v6ops/current/msg22455.html

(cc) BY NC SA

のみでApacheなどのWebサーバを起動していると思っていたら、実はIPv6側でもHTTPを受け付けていたという場合があります。これを確かめるには、Webサーバを起動している機器で、`netstat`か`lsof`か`ss`を実行してみてください。グローバルなIPv6アドレスが設定されていない機器であっても、気がつかずに`localhost(::1)`やリンクローカルアドレスでHTTPを受け付けているかもしれません。「IPv6は自分には関係がない」と思っていたとしても、気づかずにIPv6で通信可能な状態になっていることもあるので、注意が必要です。

たとえば以下の例では、特定のIPアドレスを指定せずに`AF_INET6`（IPv6）のTCPソケット（`SOCK_STREAM`）に対して`bind`を行っています。

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int
main()
{
  int sock0;
  struct sockaddr_storage client;
  socklen_t len;
  int sock;
  struct addrinfo hints, *res;
  int err;

  memset(&hints, 0, sizeof(hints));
  hints.ai_family = AF_INET6;
  hints.ai_flags = AI_PASSIVE;
  hints.ai_socktype = SOCK_STREAM;
  err = getaddrinfo(NULL, "12345", &hints, &res);
  if (err != 0) {
    printf("getaddrinfo : %s\n", gai_strerror(err));
    return 1;
  }

  /* ソケットの作成 */
  sock0 = socket(res->ai_family, res->ai_socktype, 0);
  if (sock0 < 0) {
    perror("socket");
    return 1;
  }

  if (bind(sock0, res->ai_addr, res->ai_addrlen) != 0) {
    perror("bind");
    return 1;
  }

  freeaddrinfo(res); /* addrinfo構造体を解放 */

  /* TCPクライアントからの接続要求を待てる状態にする */
  listen(sock0, 5);

  /* TCPクライアントからの接続要求を受け付ける */
  len = sizeof(client);
  sock = accept(sock0, (struct sockaddr *)&client, &len);
```

CC BY NC SA

```
    /* 6文字送信 */
    write(sock, "HELLO\n", 6);

    /* TCPセッションの終了 */
    close(sock);

    /* listen するsocketの終了 */
    close(sock0);

    return 0;
}
```

IPV6_V6ONLYがデフォルトで無効になっていない環境でこのプログラムを実行すると、IPv4のTCPで12345番ポートに接続しても、IPv6のTCPで12345番ポートに接続しても、通信ができてしまいます。試しに、telnet ::1 12345およびtelnet 127.0.0.1 12345を実行してみてください。

IPv4とIPv6の両方でTCPの受付が行われていることは、他の方法でも確認できます。たとえば、このサーバプログラムを実行中に「netstat -na」をmacOSから実行すると、次のような結果が得られます。

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp46      0      0  *.12345                *.*                    LISTEN
```

Protoの部分がtcp46となっており、IPv4とIPv6の両方で接続を受け付けていることがわかります。AF_INETで作ったソケットであれば、この部分がtcp4となります。IPV6_V6ONLYを有効にした状態のAF_INET6ソケットであればtcp6になります。

15.5　ポリシーテーブルの実装

getaddrinfo()は、複数のIPアドレスのリストを結果として返す場合があります。getaddrinfo()が返すリストの並びによって、ユーザがどのようなIPアドレスで通信を行うのかが変わってきます。getaddrinfo()が結果として返すIPアドレスの順番は、ネットワークを利用するアプリケーションにとって重要な要素です。

getaddrinfo()が結果として返すIPアドレスの順番に関しては、RFC 6724にて、デフォルトIPv6アドレス選択のルールが定められています。RFC 6724では、getaddrinfo()が返す順番を開発者が制御できるように、ポリシーテーブルという仕組みを利用可能とするように推奨しています。gai.confは、RFC 6724で定義されているポリシーテーブルの実装のひとつです。glibcを利用するシステムでは、/etc/gai.confという設定ファイルを編集することで、そのホスト内でgetaddrinfoが返す結果の優先度に影響を与えることが可能です。

RFC 6724ではポリシーテーブルの設定例がいくつか示されています。ポリシーテーブルについて詳しくは13.1.3項を参照してください。

cc BY NC SA

第III部

DNSとIPv6

直接の互換性がないIPv6とIPv4は、同一のネットワーク上で同時に利用できます（デュアルスタック）。インターネットにおいてIPv6/IPv4デュアルスタック環境を実現する鍵は名前解決です。ここでは、ユーザがIPv4とIPv6の違いを認識せずにインターネットを使えるようにするDNSについて説明します。

- DNSの基本的な仕組み
- DNSのIPv6対応で必須となるEDNS0拡張
- DNSによるIPv6/IPv4デュアルスタック環境の実現

[書籍版では空白のページです]

CC BY NC SA

第16章

DNSの基礎とIPv6対応

名前からIPアドレスを調べ出すことを「名前解決」といいます。現在のインターネットで名前解決のために採用されているのが**DNS**（Domain Name System）です。インターネットを利用するための仕組みの大半が「まず最初に名前解決を行い、それによって得られたIPアドレスを使って通信を行う」という動きをするため、DNSはIPv6にとっても非常に重要な要素です。

16.1 DNSの仕組み

DNSプロトコルの基本は、1987年に策定されたRFC 1034[†1]とRFC 1035[†2]です。最も基本となる2つのRFCが策定されたのが1987年にまで遡ることもあり、DNSの細部の表現には不明瞭な部分もあります。RFC 1034とRFC 1035の曖昧な部分などは、RFC 2181[†3]で明確化されています。さらに、RFC 1034とRFC 1035に対するさまざまな変更が別のRFCとして策定されています。また、RFC 6195[†4]には、DNSについてのIANA的な検討事項がまとめられています。

根幹的なRFCの更新は珍しい？

IPv6に関連するRFCは、上書きによる廃止がたびたび起きています。たとえば、IPv6の基本仕様を示しているRFCは、RFC 1883（1995年）→RFC 2460（1998年）→RFC 8200（2017年）のように上書き廃止による更新が行われています。

その一方で、DNSに関しては、RFC 1034とRFC 1035を基本としつつ、その周辺が拡張されていくという傾向があります。TCPの基本的な仕様を示したRFC 793も、IPv4の基本的な仕様を示したRFC 791も、両方とも1981年に発行されたものが今も標準として残っており、上書き廃止はされていません。

†1 RFC 1034：P.V. Mockapetris, “Domain names - concepts and facilities”, 1987年11月
†2 RFC 1035：P.V. Mockapetris, “Domain names - implementation and specification”, 1987年11月
†3 RFC 2181：R. Elz, R. Bush, “Clarifications to the DNS Specification”, 1997年7月
†4 RFC 6195：D. Eastlake 3rd, “Domain Name System (DNS) IANA Considerations”, 2011年3月

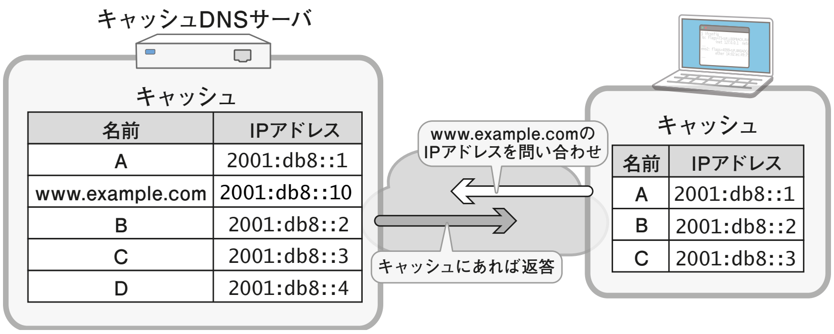
そう考えると、あまり使われていないため仕様を変更したとしても影響が少ないという背景はあったのかもしれませんが、IPv6のように「上書き廃止されつつも基本仕様の内容が更新されていく」というのは、インターネットに関連する根本的な基本仕様としては、珍しい事例なのかもしれません。

16.2　DNSサーバへの再帰問い合わせと反復問い合わせ

DNSの特徴は、名前に対応するIPアドレスをインターネットのどこか1箇所ですべて把握しているわけではない点です。さまざまなDNSサーバが各自の範囲内の情報を持っていて、その情報をたどっていくことで最終的な結果を得るというのがDNSの仕組みです。

DNSの仕組みの概観を知るために、ユーザの手もとにある機器で名前解決を行う場合を考えてみましょう。ユーザの手もとにある機器で動作するアプリケーションが、名前解決に関連する過去の記録（キャッシュといいます）を利用するように作られている場合、まずはキャッシュを確認します[†5]。もし過去にwww.example.comのIPアドレスを調べたことがあれば、それがキャッシュとして記録に残っているので、そのときに利用したIPアドレスを使います。

アプリケーションが保持するキャッシュにない名前（過去に調べたことがない名前だったり、過去に名前解決したことがあってもある程度の時間が経ってキャッシュがタイムアウトしたりしている場合）は、名前とIPアドレスの対応がたくさんキャッシュされている外部のサーバにwww.example.comの名前解決を依頼します。そのような外部のサーバを**キャッシュDNSサーバ**と呼びます（図16.1）。



▶図16.1　DNSクライアントとキャッシュDNSサーバ

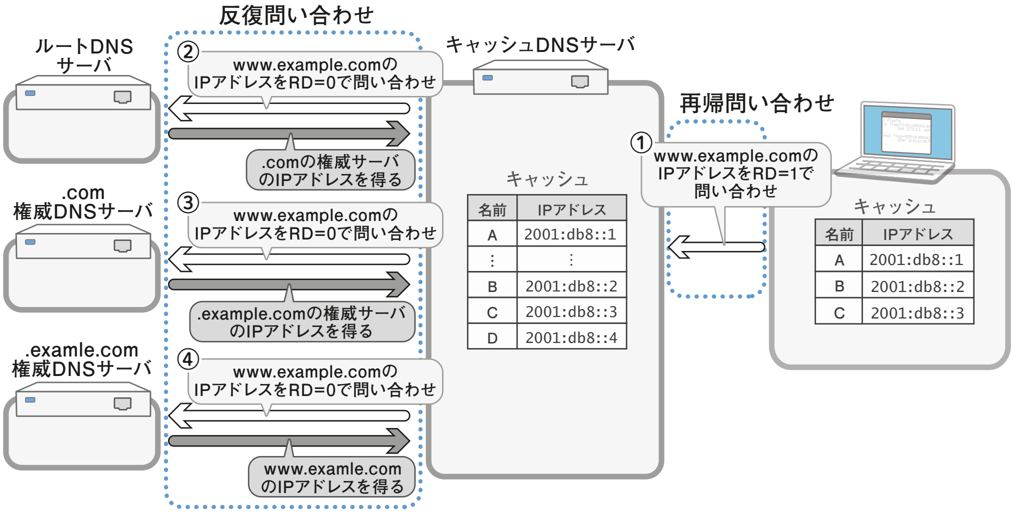
ここで注意してほしいのは、名前解決を依頼するキャッシュDNSサーバが、その名前に対応するIPアドレスを知っているとは限らない点です。キャッシュDNSサーバでは、問い合わせを受けた名前に対応するキャッシュをいっさい持ってない場合、他のDNSサーバに問い合わせを行います。DNSサーバに対して、「自分で管理している範囲内の情報で問い合わせに対する回答ができない場合には、問い合わせに対する回答が完結するまで他のDNSサーバに代理で問い合わせを行ってほしい」と依頼して問い合わせることを、**再帰問い合わせ**と呼びます。再帰問い合わせではDNSメッセージのRD（Recursive Desired）フラグを1に設定します。RD

†5　キャッシュを利用しないアプリケーションもあります。

CC BY NC SA

フラグが1に設定されたDNSメッセージを受け取ったキャッシュDNSサーバは、自分が持っていない情報を他のサーバに問い合わせます。ユーザがキャッシュDNSサーバに対して行う問い合わせは再帰問い合わせです。逆に言うと、ユーザに代わって他のDNSサーバに問い合わせを行うように設定されているのがキャッシュDNSサーバということになります。

再帰問い合わせの要求を受け取ったキャッシュDNSサーバは、他のDNSサーバに対して名前解決を問い合わせますが、このときはRDフラグが0のDNSメッセージを利用します。RDフラグが0のDNSメッセージを受け取ったDNSサーバでは、再帰問い合わせをせず、自分が知る範囲の情報のみを回答します。これを**非再帰問い合わせ**や**反復問い合わせ**と呼びます（図16.2）。



▶ 図16.2 DNSによる名前解決

キャッシュDNSサーバは、いっさいのキャッシュを持たない場合、まず**ルートDNSサーバ**と呼ばれるサーバに`www.example.com`のIPアドレスを問い合わせます。ルートDNSサーバについては、IPアドレスが事前に配布されているので、この問い合わせは常に可能です。

キャッシュDNSサーバからルートDNSサーバに`www.example.com`の名前解決の問い合わせをしても、この名前に対応するIPアドレスがすぐに判明するわけではありません。ルートDNSサーバが返すのは、`.com`の範囲の名前解決に必要な情報をすべて持っている別のサーバの名前です。その別のサーバは、いまの例であれば「`.com`という限られた範囲の名前解決についての権威」なので、**権威サーバ**と呼ばれます。キャッシュDNSサーバは、権威サーバを示す情報の付加情報（グルー）として、権威サーバのIPアドレスが付いていれば、そのIPアドレスに対して改めて`.com`の権威サーバに`www.example.com`の名前解決を依頼することになります。

NOTE

`.com`の権威DNSサーバであれば、`.com`で終わる名前を解決するための情報はすべて持っていますが、たとえば`.org`で終わる名前についての情報や、`.org`そのものについては、まったく知りません。ただし、これは`.com`以外についての情報を知っていてはいけないという意味ではなく、たとえば実際の運用でも`.com`の権威DNSサーバは`.net`の情報も知っています。

CC BY NC SA

キャッシュDNSサーバから、`www.example.com`の名前解決を依頼された`.com`の権威サーバは、`.example.com`の権威サーバの名前を返します。このとき、返答メッセージ内に権威サーバの名前に対するグルーとして権威サーバのIPアドレスが付属していれば、`example.com`の権威サーバのIPアドレスが得られます。グルーが付属していない場合には、権威サーバの名前から別途問い合わせが必要です。

権威サーバのIPアドレスを得たキャッシュDNSサーバは、今度は`.example.com`の権威サーバ`www.example.com`に対して名前解決の問い合わせを行います。`.example.com`の権威サーバは、`www.example.com`のIPアドレスを知っているので、対応するIPアドレスを返します。最終的に`www.example.com`のIPアドレスを得たキャッシュDNSサーバは、もともと名前解決を依頼していたクライアントに、その結果を通知します。

NOTE

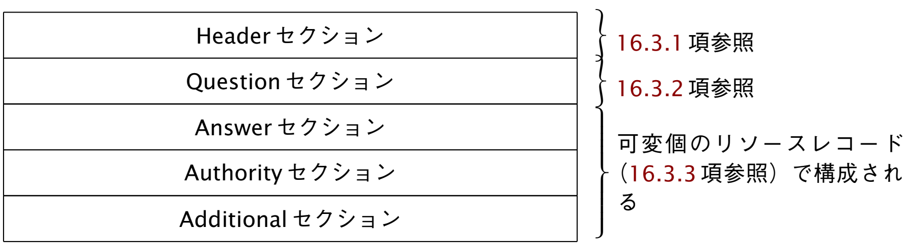
`www.example.com`のように、各部がそろってIPアドレスと結びつく「名前」のことを**FQDN**（Fully Qualified Domain Name、完全修飾ドメイン名）と呼ぶこともあります。

DNSの特徴を一言でまとめると、「各権威サーバは自分が把握すべき範囲を知っている」といえます。すべての情報をどこか1箇所で把握しているのではなく、各自が分担して自分の責任範囲を定義し、知っている範囲内で次の権威サーバを教えるのです。インターネット全体という巨大なシステムは、このような名前解決の分散管理のおかげで、滞りなく動き続けることが可能となっています。

16.3　DNSメッセージフォーマット

DNSプロトコルで名前解決の問い合わせと応答に使うメッセージは、行きも帰りも同じフォーマットをしています。DNSに関連する情報を調べるときに使う`dig`や`nslookup`といったコマンドの実行結果の意味を知るためには、このフォーマットに関する知識が必要です。

DNSメッセージのフォーマットはRFC 1035で定義されており、図16.3に示すような5つの**セクション**に分かれています。これらHeader、Question、Answer、Authority、Additionalの各セクションが、必要に応じて各種のメッセージに含まれます。

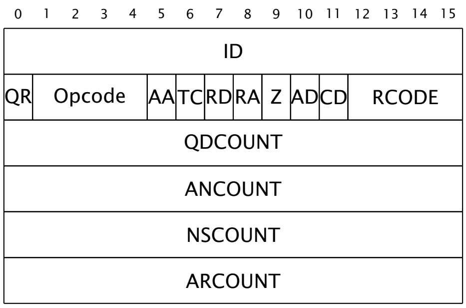


▶図16.3　DNSメッセージフォーマット

16.3.1 Headerセクション

Headerセクションは、すべてのDNSメッセージに含まれます。DNSメッセージ全体の意味を示す重要なフィールドであり、そのDNSメッセージにどんなセクションが含まれているかや、そのDNSメッセージがクエリなのか応答なのか、といった情報を含みます。

Headerセクションは以下のようなフォーマットになっています。



▶図16.4　Headerセクション

- ID（16ビット）

  このDNSメッセージのIDを示します。IDの値は問い合わせ側で生成され、応答側からの返答では、問い合わせメッセージのIDフィールドの値が応答メッセージのIDフィールドにコピーされます。

- QR（1ビット）

  このDNSメッセージがクエリ（0）であるか応答（1）であるかを示します。

- Opcode（4ビット）

  クエリの種類を示します。RFC 1035では、クエリの種類として、QUERY、IQUERY、STATUSの3種類が定義されています。ただし、この3種類のうち、IQUERYはRFC 3425[†6]で廃止されました。

  RFC 1035で定義された3種類以外には、RFC 1996[†7]で定義されたNOTIFYと、RFC 2136[†8]で定義されたUPDATEがあります。

- AA（1ビット）

  Authoritative Answerを意味し、権威DNSサーバからの応答であることを示すフラグです。

- TC（1ビット）

  TrunCationを意味するフラグです。転送可能なサイズよりもメッセージが大きくなる

†6 RFC 3425：D. Lawrence, “Obsoleting IQUERY”, 2002年11月

†7 RFC 1996：P. Vixie, “A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)”, 1996年8月

†8 RFC 2136：P. Vixie, S. Thomson, Y. Rekhter, J. Bound, “Dynamic Updates in the Domain Name System (DNS UPDATE)”, 1997年4月

ためにメッセージが切断されていることを示します。詳しくは、16.6節を参照してください。

- RD（1ビット）

Recursion Desiredを意味するフラグです。サーバは、受け取った問い合わせに対する権威ある回答を持っていない場合、その問い合わせにRDフラグがセットされていれば、再帰的に他のDNSサーバに問い合わせできます。RDフラグが1の場合は「サーバに再帰問い合わせをすることを要求する」、0の場合は「サーバに再帰問い合わせをしないように要求する」という意味になります。

- RA（1ビット）

Recursion Availableを意味するフラグです。サーバが再帰問い合わせをサポートしているかどうかを示します。

- Z（1ビット）

予約済みのビットです。ゼロにセットされます。RFC 1035では3ビットですが、RFC 4035[†9]でADとCDビットが定義されています。

- AD（1ビット）

Authentic Dataを意味するフラグです。DNSSECで使われます。RFC 4035で定義されています。

- CD（1ビット）

Checking Disabledを意味するフラグです。DNSSECで使われます。RFC 4035で定義されています。

- RCODE（4ビット）

Response Codeを意味し、応答の状態を示します。状態として設定可能な値としては、No error（0）、Format error（1）、Server failure（2）、Name Error（3）、Not Implemented（4）、Refused（5）などがあります。

- QDCOUNT（16ビット）

Questionセクションに含まれるエントリ数を示します。RFC 1035の仕様上は複数の問い合わせが同時にできるので、このビットが1より大きい場合もありえるはずですが、実装上は1以外の数値での問い合わせは正常に動作しないので注意が必要です。

- ANCOUNT（16ビット）

Answerセクションに含まれるリソースレコードの数を示します。

- NSCOUNT（16ビット）

Authorityセクションに含まれるName server resource recordの数を示します。

- ARCOUNT（16ビット）

Additionalセクションに含まれるRecource Recordの数を示します。

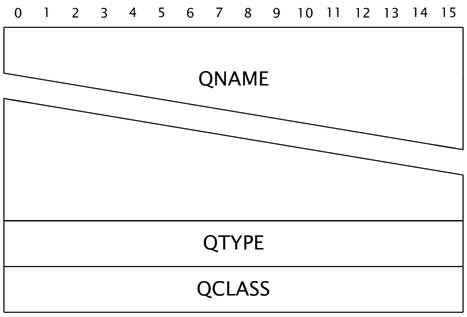
---

†9　RFC 4035：R. Arends,R. Austein,M. Larson,D. Massey,S. Rose, “Protocol Modifications for the DNS Security Extensions”, 2005年3月

CC BY NC SA

16.3.2　Question セクション

Question セクションには、DNS サーバへの問い合わせ内容を含めます。



▶ 図 16.5　Question セクション

- QNAME

　問い合わせるドメイン名を指定するフィールドです。ドメイン名は、そのままの文字列ではなく、サブドメインを表すラベル（label）の列として表現します。たとえば、問い合わせるドメイン名が `example.com` であれば、`example` を示すラベルと、`com` を示すラベルと、ルートを示すラベルとを順番に QNAME に指定することになります。

　各ラベルの先頭には、そのラベルの長さも 1 オクテットで指定します。つまり、`example` を示すラベルには 7 を、`com` を示すラベルには 3 を指定します。ルートを示すラベルは長さゼロとみなし、0 を指定します。したがって、ラベル列全体は、このルートを示すラベルの長さを意味する 0 で終端されることになります。

　例として、`test.example.com` を問い合わせる場合に QNAME に指定するラベル列は次のようになります。

	t	e	s	t		e	x	a	m	p	l	e		c	o	m	
0x04	0x74	0x65	0x73	0x74	0x07	0x65	0x78	0x61	0x6d	0x70	0x6c	0x65	0x03	0x63	0x6f	0x6d	0x00

NOTE

このラベルの形式は RFC 1035 の Section 3.1 で示されており、DNS だけでなく、名前を利用するさまざまなプロトコルでも活用されています。

　なお、QNAME 全体の長さは奇数オクテットになる場合もあります。

- QTYPE（16 ビット）

　問い合わせの種類を示すフィールドです。IPv4 アドレスを要求する A（1）や、IPv6 アドレスを要求する AAAA（28）、権威 DNS サーバに対する情報要求を意味する NS（2）などが代表的な値です。

　QTYPE フィールドでは、リソースレコードの TYPE フィールド（305 ページ参照）と同じ数値が使われます。ただし、リソースレコードの TYPE フィールドに指定できる値は、QTYPE フィールドに指定できる値のサブセットになっています。リソースレコード

cc BY NC SA

のTYPEに存在しないQTYPEとしては、ゾーン全体をリクエストするAXFR（252）、関連するすべて（ANY）のレコードを要求する「*」（255）などがあります（ANYについては16.7.3項も参照してください）。

- QCLASS（16ビット）

  RFC 1035では、後述するリソースレコードについていくつかのクラス（class）を規定しており、それを指定するフィールドです。本書執筆時点では、インターネットを意味するIN（1）という1種類のクラスのみが利用されています。IPv4でもIPv6でも、両方ともこのINを利用しています。

16.3.3　リソースレコード（Resource Record）

問い合わせに対する回答の内容は、Answer、Authority、Additionalの3種類のセクションに含めて返信されます。

- **Answerセクション**は、Questionセクションに記述された問い合わせに対する回答を示します。
- **Authorityセクション**には、ドメイン名の委任先の権威DNSサーバを示すNSレコードなどが含まれます。
- **Additionalセクション**には、付随情報として、Authorityセクションに記述された権威DNSサーバのIPv4アドレス（Aレコード）やIPv6アドレス（AAAAレコード）などが含まれます。

Answer、Authority、Additionalは、空になる場合もあります。

これら3種類のセクションは、いずれも、可変個の**リソースレコード**（**RR**）で構成されます。リソースレコードは、DNSでやり取りするドメインの性質を表すもので、セクションにおいては図16.6のようなフォーマットによって示されます。

- NAME

  リソースレコードに関連するドメイン名を示すフィールドです。

- TYPE

  リソースレコードの種類を示します。QuestionセクションのQTYPEの説明で触れたAやAAAA、NSなどが代表的なリソースレコードです。

- CLASS

  リソースレコードのクラスを示します。QCLASSと同様、通常はインターネットを示すクラスのIN（1）が利用されます。

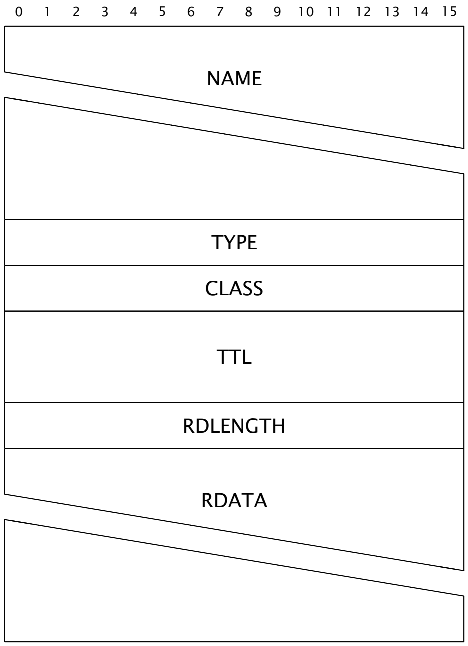
- TTL（32ビット）

  このリソースレコードをキャッシュとして保持してもよい秒数を示しています。

- RDLENGTH

  後述するRDATAの長さを示します。

CC BY NC SA



▶ 図 16.6 リソースレコード

- RDATA

  リソースを表現する可変長のフィールドです。TYPE フィールドと CLASS フィールドの組み合わせによってフォーマットが異なります。たとえば、TYPE フィールドが AAAA で CLASS フィールドが IN の場合は、128 ビットの IPv6 アドレスが格納されます。TYPE フィールドが A で、CLASS フィールドが IN の場合は、32 ビットの IPv4 アドレスが格納されます。

■ リソースレコードの TYPE フィールドに指定できる値

RFC 1035 では、リソースレコードの TYPE フィールドの値は Question セクションの QTYPE フィールドの値のサブセットであると書かれています。実際、リソースレコードの TYPE フィールドでは、QTYPE フィールドと同じ値が使われます。したがって、IPv4 アドレスの名前解決であれば A（1）、権威のあるネームサーバ情報であれば NS（2）を指定します。

本書の主題である IPv6 アドレスの名前解決の場合は、QTYPE として AAAA（28）を指定します。AAAA レコードは RFC 3596[†10] で定義されています。STUN（第 20 章参照）や ISATAP（22.3 節参照）で利用される QTYPE である SRV（33）もあります。16.5 節で解説する逆引きでは PTR（12）を指定します。

[†10] RFC 3596：S. Thomson, C. Huitema, V. Ksinant, M. Souissi, “DNS Extensions to Support IP Version 6”, 2003 年 10 月

CC BY NC SA

TYPEフィールドに指定できる値の詳細は、IANAのリスト[†11]およびRFC 6195を参照してください。

■ IPv6に関連するリソースレコードの拡張

DNSが開発された当初は、IPv4のみが前提であったため、DNSでIPv6アドレスを表現する手段がありませんでした。そのため、IPv6をDNSで表現するために新しいリソースレコードのTYPEを追加するという形で、既存のDNS仕様を拡張してIPv6を表現できるようにするRFC 1886[†12]が1995年に発行されました。その後、RFC 1886を上書き廃止するRFC 3596が2003年に発行されたので、現在ではDNSに対するIPv6拡張はRFC 3596で定義されています。

RFC 3596では、IPv6アドレスを示すAAAAレコードと、IPv6の逆引きアドレスを示す`IP6.ARPA`ドメインが定義されています[†13]。

IPv6アドレスを示すAAAAレコードのデータ部分は、128ビットのIPv6アドレスがネットワークバイトオーダーで表現されます。

IPv6の逆引きのアドレスは、下位ビットから先に4ビット区切りで並べた16進数で表現されます。たとえば、`2001:db8:1:2:3:4:5:6`というIPv6逆引きアドレスは、以下のようになります。

```
6.0.0.0.5.0.0.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA
```

NSやMXなど、AdditionalセクションでAレコードと関連づけられるリソースレコードについては、従来のAレコードだけではなく、AAAAレコードとも関連づけられるようにする必要があります。そういったDNSの機能に必要な変更についても、IPv6対応としてRFC 3596に含まれています。

16.4　IPv4とIPv6アドレスの問い合わせ例

フォーマットだけを見ていてもわかりにくいので、IPv4アドレスとIPv6アドレスの名前解決を行ったときのDNSメッセージの例を見てみましょう。ここでは、ユーザとキャッシュDNSサーバ間のやり取りの例を紹介します。ユーザがキャッシュDNSサーバに対して`www.example.com`の名前解決を行う場合を考えます。

ユーザがキャッシュDNSサーバに対し、`www.example.com`のAレコードとAAAAレコードの問い合わせを別々のセッションとして送信したとします。キャッシュDNSサーバは、2つの異なる問い合わせに対して、それぞれ応答します。

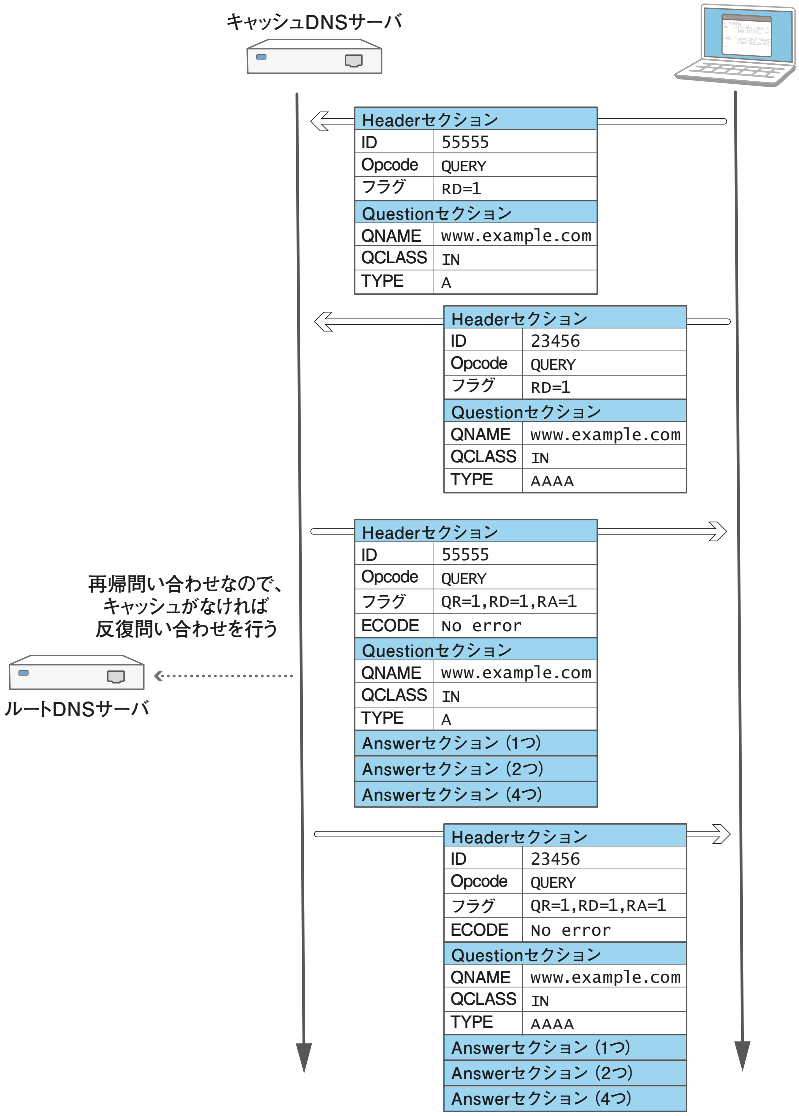
図16.7の例では、IDが`55555`のDNSメッセージで`www.example.com`のAレコードを問い合わせ、IDが`23456`のDNSメッセージで`www.example.com`のAAAAレコードを問い合わせています。問い合わせを受け取ったキャッシュDNSサーバにキャッシュがあれば、そのまま結

†11　Resource Record (RR) TYPEs ：
https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4

†12　RFC 1886：S. Thomson,C. Huitema, “DNS Extensions to support IP version 6”, 1995年12月

†13　IPv4の逆引きには`in-addr.arpa`が利用されています。

CC BY NC SA



▶ 図16.7 www.example.comの名前解決例

果を返しますが、キャッシュがなければキャッシュDNSサーバによって反復問い合わせが行われます。反復問い合わせが行われるのは、ユーザからのDNSメッセージのHeaderセクションでRDビットが1になっているためです。

キャッシュDNSサーバからユーザへの返答を見ると、HeaderセクションのRCODEが0でエラーなしとあり、問い合わせが成功していることを示しています。次に、Headerセクションのフラグを見ると、QRおよびRAビットが1となっています。QRビットが1となっているのは、そのDNSメッセージが問い合わせに対する回答であるためです。RAビットが1となっているのは、キャッシュDNSサーバが再帰問い合わせをサポートしていることを示しています。

返答パケットのAnswerセクションは1つですが、そこには`www.example.com`のIPv4アドレスが含まれています。Authorityセクションは2つありますが、それらには`www.example.`

comの権威DNSサーバのFQDNであるa.iana-servers.netとb.iana-servers.netが示されています。Additionalセクションは4つありますが、a.iana-servers.netのIPv4アドレスを示すAレコードとa.iana-servers.netのIPv6を示すAAAAレコード、b.iana-servers.netのIPv4アドレスを示すAレコードとb.iana-servers.netのIPv6アドレスを示すAAAAレコードの4つとなっています。

AAAAレコードを問い合わせたIDが23456のDNSメッセージも、基本的にAレコードと同じような結果となっています。

16.4.1 Aのdig結果

キャッシュDNSサーバとやり取りされるDNSメッセージの内容をdigコマンドの結果で確かめてみましょう。dig a www.example.com.の実行結果は、たとえば次のようになります。

```
; <<>> DiG 9.8.3-P2 <<>> a www.example.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63333
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.example.com.       IN  A

;; ANSWER SECTION:
www.example.com.    51298   IN  A   192.0.43.10

;; AUTHORITY SECTION:
example.com.        8262    IN  NS  b.iana-servers.net.
example.com.        8262    IN  NS  a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net. 1800    IN  A   199.43.132.53
a.iana-servers.net. 1800    IN  AAAA    2001:500:8c::53
b.iana-servers.net. 1800    IN  A   199.43.133.53
b.iana-servers.net. 1800    IN  AAAA    2001:500:8d::53

;; Query time: 17 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Fri Sep 14 17:12:11 2012
;; MSG SIZE  rcvd: 185
```

16.4.2 AAAAのdig結果

dig aaaa www.example.com.の実行結果は、たとえば次のようになります。

```
; <<>> DiG 9.8.3-P2 <<>> aaaa www.example.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55140
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.example.com.       IN  AAAA
```

CC BY NC SA

```
;; ANSWER SECTION:
www.example.com.    172800  IN  AAAA    2001:500:88:200::10

;; AUTHORITY SECTION:
example.com.        8250    IN  NS  b.iana-servers.net.
example.com.        8250    IN  NS  a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net. 1788    IN  A   199.43.132.53
a.iana-servers.net. 1788    IN  AAAA    2001:500:8c::53
b.iana-servers.net. 1788    IN  A   199.43.133.53
b.iana-servers.net. 1788    IN  AAAA    2001:500:8d::53

;; Query time: 17 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Fri Sep 14 17:12:11 2012
;; MSG SIZE  rcvd: 197
```

16.5 DNSの逆引き

ここまでの説明では、DNSについて、名前からIPアドレスを解決する手段として説明してきました。DNSの実装では、これとは逆、つまりIPアドレスから名前を検索する機能も用意されています。これをDNSの逆引きといいます。DNSの逆引きの話題では、名前からIPアドレスを解決することを「正引き」と表現することがあります。

詳しくは16.5.1項で説明しますが、DNSの逆引きは、逆引きを行うための特殊な名前にIPアドレスを変換したうえで名前解決を行います。

IPv4を利用したこれまでのインターネットでは、さまざまな場面でDNSの逆引きが利用されることがあります。たとえばWebサーバでは、接続相手の情報をアクセスログに書き込む際、IPアドレスではなく逆引き後の名前を記述するように設定できます。`sshd`や`inetd`のよくある初期設定でも、接続受付時に逆引きが行われます。逆引き設定が存在しないと、メールサーバがメールを受け取れない場合もあります。逆引き設定が存在しないと接続処理に時間がかかる可能性もあります。

注意が必要なのは、名前からIPアドレスの名前解決である正引きの結果と、逆引きの結果が一致するとは限らない点です。逆引きは、あくまでリソースレコードのひとつであり、正引きとの整合性が必ず取れるものとは限らないのです。そのため、逆引きは、参考にする情報のひとつという程度であり、逆引きで設定されている内容そのものが何かを保証するものでもないのです。

16.5.1 逆引きの仕組み

逆引きは、IPアドレスを逆に並べた特殊な名前を利用して行われます。IPv4では`.in-addr.arpa.`が利用され、IPv6では`.ip6.arpa.`が利用されます。

たとえば、`192.0.2.3`というIPv4アドレスに対する逆引きを行うときには、`3.2.0.192.in-addr.arpa.`という名前を利用して逆引きを行います。`2001:db8::1:2:3456`というIPv6アドレスの場合には、`6.5.4.3.2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.`という名前を利用して逆引きが行われます。

CC BY NC SA

逆引きに利用されるQTYPEおよびリソースレコードのTYPEはPTRです。PTRレコードは、逆引きに対して対応する名前を返します。

IPv4とIPv6における逆引きのためのトップレベルドメインの運用に関しては、RFC 5855[†14]（BCP 155）で説明されています。

RFC 5855では、`.in-addr.arpa.`のためのNSは次のような名前になるとあります。

```
A.IN-ADDR-SERVERS.ARPA
B.IN-ADDR-SERVERS.ARPA
C.IN-ADDR-SERVERS.ARPA
D.IN-ADDR-SERVERS.ARPA
E.IN-ADDR-SERVERS.ARPA
F.IN-ADDR-SERVERS.ARPA
...
```

RFC 5855では上記のようにAからFまでの名前が例として示されているものの、6個に限定しないために最後が「`...`」になっています。とはいえ、本書執筆現在、`dig ns in-addr.arpa.`の結果はaからfになっています。

同様に、RFC 5855ではIPv6用の`.ip6.arpa.`のためのNSは次のような名前になるとあります。やはり、本書執筆現在、`dig ns ip6.arpa.`の結果はaからfになっています。

```
A.IP6-SERVERS.ARPA
B.IP6-SERVERS.ARPA
C.IP6-SERVERS.ARPA
D.IP6-SERVERS.ARPA
E.IP6-SERVERS.ARPA
F.IP6-SERVERS.ARPA
...
```

ARPAトップレベルドメインは、作られた当初は、インターネットの前身であるARPANETに関連するものでしたが、現在は「Address and Routing Parameters Area」の略であるとされています。

IPv6の逆引きで利用される名前は、かつては`ip6.int`でしたが、いまは`ip6.int`は廃止されています。`ip6.int`に関しては16.8.1項を参照してください。

16.5.2　digによるIPv4アドレス逆引きの例

逆引きについても`dig`コマンドによる例を見ていきましょう。まずは、IPv4アドレス逆引きの例を紹介します。

`dig`コマンドでは、引数`-x`を利用することで逆引きができます。IPv4アドレスに対する逆引き`dig -x 198.41.0.4`の実行結果は、たとえば次のようになります。

```
; <<>> DiG 9.8.3-P2 <<>> -x 198.41.0.4
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37764
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

---
†14　RFC 5855：J. Abley, T. Manderson, “Nameservers for IPv4 and IPv6 Reverse Zones”, 2010年5月

cc BY NC SA

```
;; QUESTION SECTION:
;4.0.41.198.in-addr.arpa.        IN      PTR

;; ANSWER SECTION:
4.0.41.198.in-addr.arpa. 900     IN      PTR     a.root-servers.net.

;; Query time: 21 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Thu Sep 14 17:16:57 2017
;; MSG SIZE  rcvd: 115
```

この例では、AルートサーバのIPv4アドレスの逆引きを行っています。Questionセクションは、198.41.0.4というIPv4アドレスから計算されたin-addr.arpaの名前と、CLASSフィールドがIN、QTYPEフィールドがPTRです。

Questionに対するAnswerセクションには、Questionセクションに示されたIPv4アドレスに対応する名前がa.root-servers.net.であるという回答が含まれています。

16.5.3 digによるIPv6アドレス逆引きの例

次は、IPv6アドレスの逆引きの例です。

digコマンドでは、引数-xを利用することで逆引きができます。IPv6アドレスに対する逆引きdig -x 2001:503:ba3e::2:30の実行結果は、たとえば次のようになります。

```
; <<>> DiG 9.8.3-P2 <<>> -x 2001:503:ba3e::2:30
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13211
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;0.3.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.e.3.a.b.3.0.5.0.1.0.0.2.ip6.arpa. IN PTR

;; ANSWER SECTION:
0.3.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.e.3.a.b.3.0.5.0.1.0.0.2.ip6.arpa.
↪  86400       IN PTR a.root-servers.net.

;; Query time: 19 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Thu Sep 14 17:21:34 2017
;; MSG SIZE  rcvd: 161
```

この例では、AルートサーバのIPv6アドレスの逆引きを行っています。Questionセクションは、2001:503:ba3e::2:30というIPv6アドレスから計算されたip6.arpaの名前と、CLASSがIN、QTYPEがPTRです。

Questionに対するAnswerセクションには、Questionセクションに示されたIPv6アドレスに対応する名前がa.root-servers.net.であるという回答が含まれています。

16.5.4 IPv6逆引き問題

さまざまなサービスに影響を与えるDNSの逆引きですが、IPv6ではすべてのアドレスに対して逆引き設定を行うのが一般には困難であり、結果として「逆引き設定を行えない」という

CC BY NC SA

状況が発生します。

IPv4では、ユーザが利用するIPアドレスが限られているので、利用される可能性があるすべてのIPv4アドレスに対して逆引きの設定ができます。しかし各ネットワークセグメントごとに/64を割り当てるIPv6では、IPv4と同様に事前にすべての逆引きアドレスを設定することは非現実的です。各ユーザが64ビット空間を利用して各自のIPv6アドレスを決定するので、可能性があるすべてのIPv6アドレスを記述することによる逆引き設定が困難というわけです。

IPv6における逆引きDNSに関してはRFC 8501[†15]で問題がまとめられています。RFC 8501は、ISPにおける逆引きDNSの管理という視点のInformationalなRFCです。そこでは、/48のネットワークでの逆引き設定を考察することで、IPv4と同様にすべてのIPv6アドレスに対する逆引き設定を事前に行うことが非現実的であることが説明されています。/48のネットワークでは2の80乗個のIPv6アドレスが存在するので、そのすべてに対応する設定をゾーンに記載しようとすれば、1秒間に1000エントリをゾーンに記載していったとしても38兆年かかることになります。

もちろん、逆引き設定が可能な場合もあります。たとえば、企業内でのIPv6利用で、「システム運用部門が許可を出したマシン以外はネットワークにつながせない」という運用も可能です。その場合はハードウェアに関する情報を事前に調査可能なので、必要な部分は概ね逆引きが動くようにできます。

それ以外の場合、たとえばISPが提供する個人向けIPv6サービスなどでは、ISPが/64のアドレス空間すべてを逆引き設定できるわけではないと考えられるので、そもそも逆引きができない状況が多いでしょう。したがって逆引き設定が存在せず、それが原因で利用できないサービスが発生する可能性があります。実際、現在のインターネットでは逆引き情報を運用で要求するサービスも少なくありません。RFC 8501では、メールサーバのほか、SSHやtracerouteなどが逆引き情報を利用するアプリケーションの例として紹介されています。また、ログファイルで逆引きが利用されている件にも触れられています。

RFC 8501では、ISPがIPv6における逆引きにどう対応すればよいか、次のように整理されています。

- 不在応答（Negative Response）を返す方法

  ユーザが利用しているIPv6アドレスに対応するPTRへの問い合わせに対してNXDOMAINを返すことをISPが選択する場合が考えられます。これは合理的ではありますが、WebサービスやSSHなどでタイムアウトが発生する可能性もあります。その一方で、メールサーバでの逆引きが失敗することでメール配送が拒否されるので、スパムメールの抑止になる可能性もあります。

- ワイルドカードを利用する方法

  RFC 4472では、RFC 4592で規定されているワイルドカードをIPv6の逆引きで利用する手法が示されています。これには、正引きと逆引きの結果が一致しなくなる可能性があります。RFC 8501では、/48、/56、/64などのアドレスレンジに対するワイルドカード

[†15] RFC 8501：L. Howard, “Reverse DNS in IPv6 for Internet Service Providers”, 2018年11月

CC BY NC SA

での指定をした場合に、そのアドレスレンジに含まれる全アドレスがマッチしてしまう点が指摘されています。

- ダイナミック DNS を利用する方法

  正引きと逆引きの整合性を確保する手法として、ダイナミック DNS が利用できます。RFC 8501 では、それによる DoS 攻撃の懸念、ユーザが各自でダイナミック DNS を利用する場合の課題、ゲートウェイがダイナミック DNS を利用する場合の課題などが指摘されています。

- 逆引き DNS をユーザに委譲する方法

  ISP のユーザが自分たちで DNS サーバを運用できるなら、逆引き DNS ゾーンをユーザに委譲するのがよいでしょう。ただし、この方法が可能なのは DNS サーバの運用経験があるユーザに限られます。

- オンデマンドで PTR レコードを生成する方法

  問い合わせのたびに動的に PTR レコードを生成する方法もあります。ただし RFC 8501 では、この方法には DDoS 攻撃に関する懸念があることや、DNSSEC によるレコードを自動生成する場合には負荷が高くなる可能性などが指摘されています。

「逆引きが困難な可能性がある」という IPv6 の状況では、これまでの IPv4 の世界とは異なるサービス運用が必要になる場合もあります。IPv6 逆引きに関しては、これからさまざまな提案が登場すると考えられますが、いまのところデファクトな方式は定まっていないので、動向を見守る必要があると思われます。

16.6 DNS メッセージの 512 オクテット問題

16.3 節で説明した DNS メッセージは、通常は UDP で送信されます。このときの DNS メッセージの長さは、DNS の実装について規定した RFC 1035[†16] にて、512 オクテットまでに制限されています。

時代の変化とともに、DNS メッセージでさまざまな情報を提供するような提案が登場し、DNS メッセージの長さが 512 オクテットを超えるような状況も発生するようになりました。もちろん、DNS メッセージの Header セクションに TC ビットが用意されていることからわかるように、512 オクテットを超える DNS メッセージがまったく扱えないわけではありません。しかし、インターネット上には 512 オクテット以上の DNS データを処理できない機器もあり、問題を発生させることもあります。

本書執筆現在、512 オクテットを超える DNS メッセージを送る方法としては、下記の 2 種類があります。

- EDNS0（Extension Mechanisms for DNS）
- TCP フォールバック

IPv6 も、IP アドレス長が IPv4 より長いことから、DNS メッセージが 512 オクテットを超える

†16 RFC 1035：P.V. Mockapetris, “Domain names - implementation and specification”, 1987 年 11 月

CC BY NC SA

主な要因となっています。そのため、RFC 3226[†17]では、DNSをIPv6対応するためにEDNS0が必須であると記述されています。

16.6.1 EDNS0

EDNS0の「E」は、Extension（拡張）を意味します。最後の「0」はプロトコルバージョンを表す番号です。

EDNS0は、1999年に、RFC 2671[†18]として最初に標準化されました。このRFC 2671は、RFC 2673[†19]とともに、2013年4月にRFC 6891[†20]によって廃止されています。

EDNS0が定義している拡張機能は以下の3つです。

- RCODEの拡張
- ラベル型の拡張
- 最大512オクテットとなっていたDNSメッセージの長さを65535オクテットまで拡張

IPv6が関連するのは、上記のうち3つめの、メッセージサイズの拡張です。本書ではそれを中心に解説します。

■ EDNS0オプションを指定した場合としない場合の違いの例

EDNS0の動作を説明する前に、EDNS0が有効になっている場合となっていない場合を実際に`dig`コマンドで実験して確かめてみましょう。まず、EDNS0オプションを指定しない問い合わせの例を下記に示します。この例では`dig`コマンドを利用してMルートサーバに対して問い合わせを行っています。

```
% dig @m.root-servers.net . NS +norec +noedns

; <<>> DiG 9.8.3-P1 <<>> @m.root-servers.net . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62334
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 15

;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
```

[†17] RFC 3226：O. Gudmundsson, “DNSSEC and IPv6 A6 aware server/resolver message size requirements”, 2001年12月
[†18] RFC 2671：P. Vixie, “Extension Mechanisms for DNS (EDNS0)”, 1999年8月
[†19] RFC 2673：M. Crawford, “Binary Labels in the Domain Name System”, 1999年8月
[†20] RFC 6891：J. Damas, M. Graff, P. Vixie, “Extension Mechanisms for DNS (EDNS(0))”, 2013年4月

CC BY NC SA

```
.                        518400      IN      NS      e.root-servers.net.
.                        518400      IN      NS      j.root-servers.net.
.                        518400      IN      NS      m.root-servers.net.
.                        518400      IN      NS      f.root-servers.net.
.                        518400      IN      NS      h.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.      3600000     IN      A       198.41.0.4
b.root-servers.net.      3600000     IN      A       199.9.14.201
c.root-servers.net.      3600000     IN      A       192.33.4.12
d.root-servers.net.      3600000     IN      A       199.7.91.13
e.root-servers.net.      3600000     IN      A       192.203.230.10
f.root-servers.net.      3600000     IN      A       192.5.5.241
g.root-servers.net.      3600000     IN      A       192.112.36.4
h.root-servers.net.      3600000     IN      A       198.97.190.53
i.root-servers.net.      3600000     IN      A       192.36.148.17
j.root-servers.net.      3600000     IN      A       192.58.128.30
k.root-servers.net.      3600000     IN      A       193.0.14.129
l.root-servers.net.      3600000     IN      A       199.7.83.42
m.root-servers.net.      3600000     IN      A       202.12.27.33
a.root-servers.net.      3600000     IN      AAAA     2001:503:ba3e::2:30
b.root-servers.net.      3600000     IN      AAAA     2001:500:200::b

;; Query time: 37 msec
;; SERVER: 202.12.27.33#53(202.12.27.33)
;; WHEN: Fri Feb 23 21:31:55 2018
;; MSG SIZE  rcvd: 492
```

次は、EDNS0 オプションを設定した問い合わせの例です。dig コマンドで +bufsize=1024 というオプションを指定することにより、1024 オクテットまで受け取れるということを M ルートサーバに伝えています。

```
% dig @m.root-servers.net . NS +norec +bufsize=1024

; <<>> DiG 9.8.3-P1 <<>> @m.root-servers.net . NS +norec +bufsize=1024
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36373
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                                 IN      NS

;; ANSWER SECTION:
.                        518400      IN      NS      k.root-servers.net.
.                        518400      IN      NS      l.root-servers.net.
.                        518400      IN      NS      e.root-servers.net.
.                        518400      IN      NS      g.root-servers.net.
.                        518400      IN      NS      a.root-servers.net.
.                        518400      IN      NS      h.root-servers.net.
.                        518400      IN      NS      i.root-servers.net.
.                        518400      IN      NS      f.root-servers.net.
.                        518400      IN      NS      c.root-servers.net.
.                        518400      IN      NS      d.root-servers.net.
.                        518400      IN      NS      j.root-servers.net.
.                        518400      IN      NS      b.root-servers.net.
```

```
.                         518400      IN      NS      m.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.        3600000     IN      A       198.41.0.4
b.root-servers.net.        3600000     IN      A       199.9.14.201
c.root-servers.net.        3600000     IN      A       192.33.4.12
d.root-servers.net.        3600000     IN      A       199.7.91.13
e.root-servers.net.        3600000     IN      A       192.203.230.10
f.root-servers.net.        3600000     IN      A       192.5.5.241
g.root-servers.net.        3600000     IN      A       192.112.36.4
h.root-servers.net.        3600000     IN      A       198.97.190.53
i.root-servers.net.        3600000     IN      A       192.36.148.17
j.root-servers.net.        3600000     IN      A       192.58.128.30
k.root-servers.net.        3600000     IN      A       193.0.14.129
l.root-servers.net.        3600000     IN      A       199.7.83.42
m.root-servers.net.        3600000     IN      A       202.12.27.33
a.root-servers.net.        3600000     IN      AAAA     2001:503:ba3e::2:30
b.root-servers.net.        3600000     IN      AAAA     2001:500:200::b
c.root-servers.net.        3600000     IN      AAAA     2001:500:2::c
d.root-servers.net.        3600000     IN      AAAA     2001:500:2d::d
e.root-servers.net.        3600000     IN      AAAA     2001:500:a8::e
f.root-servers.net.        3600000     IN      AAAA     2001:500:2f::f
g.root-servers.net.        3600000     IN      AAAA     2001:500:12::d0d
h.root-servers.net.        3600000     IN      AAAA     2001:500:1::53
i.root-servers.net.        3600000     IN      AAAA     2001:7fe::53
j.root-servers.net.        3600000     IN      AAAA     2001:503:c27::2:30
k.root-servers.net.        3600000     IN      AAAA     2001:7fd::1
l.root-servers.net.        3600000     IN      AAAA     2001:500:9f::42
m.root-servers.net.        3600000     IN      AAAA     2001:dc3::35

;; Query time: 13 msec
;; SERVER: 202.12.27.33#53(202.12.27.33)
;; WHEN: Fri Feb 23 21:34:55 2018
;; MSG SIZE  rcvd: 811
```

EDNS0オプションを指定しない例ではADDITIONALの部分が15でしたが、指定した場合は27に増えています。それに伴い、512よりも小さい値であった492が811になり、512よりも大きな値になっていることがわかります。

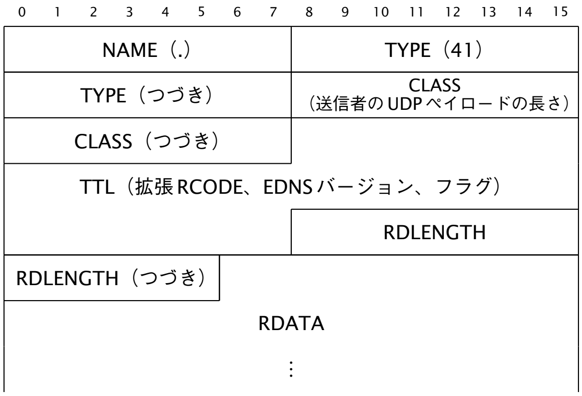
なお、この27という数字は、EDNS0の疑似レコードを含んでいます。digの結果として表示されているAdditionalセクションの項目数が26である一方で、Additional数が27となっているのは、そのためです。

■ EDNS0の疑似レコード

EDNS0では、疑似レコード（Pseudo Resource Record）を利用することで、問い合わせ先がEDNS0に対応しているかどうかを確かめてから問い合わせを行います。疑似レコードのフォーマットを図16.8に示します。図16.6に示したリソースレコードのフォーマットを踏襲していますが、リソースレコードとは根本的に用途が異なるので注意してください。

EDNS0では、NAMEフィールドに、ルートを示す「.」が入ります。TYPEフィールドとしては疑似レコードを示すOPT（41）が利用されます。CLASSフィールド（16ビット）には送信者のUDPペイロードサイズが入ります。TTLフィールドは、拡張RCODEおよびEDNSバージョンとフラグを示します。RDLENGTHフィールドは、RDATAのオクテット数です。RDATAは可変長の付属データになります。

CC BY NC SA



▶ 図16.8　EDNS0の疑似レコード

■ EDNS0の最大メッセージ長

EDNS0で指定できるメッセージ長の最大値は、16ビットで表現できる値として最大の65535です。ただし、IPv4とUDPを利用する場合には、実際に利用可能な最大のメッセージ長は65507（65535 – 20 – 8 = 65507）になります。これは、IPv4ヘッダとUDPヘッダにおけるデータ長フィールドの値には各ヘッダ自身の長さが含まれるからです。

IPv6 Jumbogram（95ページのコラム参照）を利用すれば、1つのIPv6 UDPパケットで、理論上は約4Gオクテットのデータを送信可能になります。したがって、EDNS0を利用すれば、1つのIPv6 UDPパケットで65535オクテットのデータを送信可能になります。

16.6.2　TCPフォールバック

DNSサーバは、クライアントからの要求に対する応答が512オクテットを超える場合、メッセージにTCフラグを設定して回答を送信します。クライアントは、TCフラグが設定された応答を受け取ると、改めてTCPによる問い合わせを試みます。これをDNSメッセージのTCPフォールバックと呼んでいます。DNSへの問い合わせにはUDPを使うという印象があるかもしれませんが、メッセージが512オクテットより長い場合にはTCPが利用されることがあるのです。

DNSメッセージのTCPフォールバックはEDNS0よりも昔から存在しています。1989年に発行されたRFC 1123[†21]では、DNSのTCPサポートを「SHOULD」として記述していました（RFC 1123のSection 6.1.3.2参照）。その後、2010年に発行されたRFC 5966[†22]では、DNSのTCPサポートが必須となりました。

EDNS0とTCPフォールバックの大きな違いは、EDNS0では指定できるメッセージ長の最大値が65535に制限される点です。65535よりも多いデータを送信する必要がある場合にはTCPを利用することになります。

DNSメッセージがTCPによってやり取りされる可能性に気づかず、ファイアウォールなど

---

†21　RFC 1123：R. Braden, “Requirements for Internet Hosts - Application and Support”, 1989年10月
†22　RFC 5966：R. Bellis, “DNS Transport over TCP - Implementation Requirements”, 2010年8月

cc BY NC SA

でUDPの53番ポートのみを許可してTCPの53番ポートを破棄する設定にしてしまうと、TCPフォールバックが機能しません。DNSでIPv6に関する情報を扱う必要があって応答メッセージが512オクテットを大きく超えてしまい、EDNS0ではなくTCPフォールバックを行うという状況になることもありえます。その場合、TCPの53番ポートがブロックされているとIPv6アドレスの名前解決ができなくなってしまうことになるので注意が必要です。

■ TCPフォールバック例

EDNS0オプションをクライアント側から指定せずにTCPフォールバックした例を以下に示します。この例では、Mルートサーバに対して「.」に関するANYを要求しています。ANYに対する問い合わせを答えるサーバであれば、ANYへの応答として複数のリソースレコードを回答する場合があり、その結果として応答のサイズが大きくなり、TCPフォールバックが発生することが期待できるため、ANYを使っています（ANYについては16.7.3項を参照してください）。

```
% dig @m.root-servers.net . ANY +norec
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.4.3-P3 <<>> @m.root-servers.net . ANY +norec
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42599
;; flags: qr aa; QUERY: 1, ANSWER: 21, AUTHORITY: 0, ADDITIONAL: 22

---途中省略---

;; Query time: 12 msec
;; SERVER: 202.12.27.33#53(202.12.27.33)
;; WHEN: Wed Aug 29 16:41:00 2012
;; MSG SIZE  rcvd: 1951
```

この例ではクライアント側からEDNS0オプションが設定されていないので、TCPへとフォールバックしています。

digコマンドでのMルートサーバに対して行われた、「.」にするANYの問い合わせに対し、応答の長さが1951と512を超えています。digコマンドを実行した直後に表示されている「;; Truncated, retrying in TCP mode.」という表示からは、一度UDPで問い合わせを行ったものの応答のサイズが大き過ぎたのでTCPモードでの再接続を行ったことがわかります。

16.6.3　最初からTCPでの問い合わせを行うことも可能に

RFC 1123では、ゾーン転送を除いて、DNSメッセージのやり取りではUDPに対応する必要があると記述されています。さらに、ゾーン転送でないDNSサーバへの問い合わせはまずUDPで行わなければならず、そこで得られた応答がUDPのDNSメッセージで収まりきらない大きさ（truncated）であることを示すTCビットが立っている場合にTCPでのDNSクエリを試すべきとされています。つまり、DNSサーバへの問い合わせはUDPの53番で行うべきものとされており、最初からTCPでDNSサーバへの問い合わせを行うことはRFC 1123に違反していたのです。

CC BY NC SA

この状況は、2016年3月にRFC 7766[†23]が発行されたことで変わりました。RFC 7766では、DNSサーバへの問い合わせをUDPから必ず始めるという従来の規定を緩和し、UDPよりも先にTCPを利用してもよいとしています。さらに、UDPが使えない場合の代替手段としてTCPを捉えるのではなく、有効なDNSメッセージのトランスポートとして捉えるべきであると記述されています。RFC 7766によって、DNSサーバへの問い合わせをTCPだけで行えるようになったのです。

16.7 IPv6環境におけるDNSの運用上の注意点

IPv6とDNSに関して、運用上の注意点がRFC 4472[†24]で紹介されています。

以下、そこで紹介されている注意点をいくつか紹介します。

16.7.1 特殊なIPv6アドレスの登録に関して

RFC 4472では、リンクローカルユニキャストアドレスをDNSサーバに登録すべきではないとしています。リンクローカルユニキャストアドレスはリンク内においてのみ有効であるため、正引きまたは逆引きのレコードとして登録されるべきではないとあります。

プライバシーアドレスとも呼ばれるTemporary IPv6アドレスをDNSサーバに登録することも、RFC 4472では推奨しないとあります。Temporary IPv6アドレスが変更されるたびにDNSサーバに対して最新のAAAAレコードが更新されてしまうと、本来の目的にそぐわなくなるためです。

16.7.2 IPv6対応が終わってからDNSサーバにAAAAレコードを登録すること

RFC 4472では、AAAAレコードをDNSサーバに登録するタイミングとして、以下のものをすべて満たしたときとすることを推奨しています。

- ノードのネットワークインターフェースにIPv6アドレスが割り当てられている
- 割り当てられたIPv6アドレスが設定されている
- IPv6インフラに接続されたリンクにIPv6アドレスが設定されたネットワークインターフェースが接続されている

IPv6対応が行われた一方で、グローバルなIPv6インターネットとの接続が完了していないデュアルスタックノードに関しては、DNSサーバにAAAAレコードを登録しないことによって、IPv4アドレスのAレコードのみが結果として返されるので、通信できないIPv6アドレスへの通信を試みることがありません。

---

†23 RFC 7766：J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, D. Wessels, “DNS Transport over TCP - Implementation Requirements”, 2016年3月

†24 RFC 4472：A. Durand, J. Ihren, P. Savola, “Operational Considerations and Issues with IPv6 DNS”, 2006年4月

16.7.3 QTYPE ANYでIPv4とIPv6の同時解決はできるか

RFC 1035では、QTYPEの値255を「*」とし、その内容を “A request for all records” としています。「*」はANYと表現されることも多く、デバッグ目的などで使われることがありました。ANYは、Aレコードなどの通常のリソースレコードとは異なり、問い合わせた名前に関連する複数のレコードを同時に要求するといった特殊なものです。

しかし、実際にはANYをリクエストしたからといって、そのリクエストを受け取ったDNSサーバが「すべて」を返すわけではありません。ANYに対する応答に関してはRFC 8482[†25]を参照してください。

DNSサーバがAレコードとAAAAレコードの両方を保持している状況で、そのDNSサーバにANYを含むQuestionセクションを送信すれば、AレコードとAAAAレコードが両方同時に返答される可能性はあります。しかし、ここで注意が必要なのは、ANYの定義が「対象となるサーバが保持している情報」であるという点です。キャッシュDNSサーバに対し、QTYPEとしてANYを指定したQuestionセクションを含むDNSメッセージを送信すれば、そのキャッシュDNSサーバは、そのメッセージを受け取った時点で保持している情報だけを返します。言い換えると、権威DNSサーバにAレコードとAAAAレコードの両方が登録されていたとしても、それが両方ともキャッシュDNSサーバから得られるとは限りません。このように、ANYだけで、名前に関連づけされたIPv4アドレスとIPv6アドレスの両方を1つのDNSメッセージで問い合わせたとしても、必要な結果を得られないこともあります。RFC 4472では、QTYPE=*は一般的にはデバッグや運用目的でのみ利用されるとあります。

一般に、IPv4とIPv6のデュアルスタック環境で名前解決をするときには、Aレコードに対するQuestionと、AAAAレコードに対するQuestionを、両方とも個別に送信します。QTYPEとしてANYを使えば両方を同時に得られると思うかもしれませんが、本書執筆段階ではIPv4とIPv6の名前解決を別々のDNS問い合わせで行わざるを得ない状況なのです。

16.7.4 再帰問い合わせとトランスポートの関係

RFC 4472で参照されているRFC 3901[†26]（BCP 91）では、再帰問い合わせに応じるDNSサーバがIPv6にしか対応していない状態は好ましくないとしています。ユーザに代わって名前解決を行うDNSサーバがIPv6のみに対応しているということは、IPv4で運用されているDNSサーバからゾーン情報を得られないためです。「再帰問い合わせを受けるDNSサーバは、IPv4のみ、もしくはIPv4とIPv6のデュアルスタックであること」をRFC 3901は推奨しているのです。

RFC 3901は、DNSゾーンの管理を行うDNSサーバについて、最低でも1つはIPv4によって通信可能な権威DNSサーバを運用することも推奨しています。DNSゾーンのすべての権威DNSサーバがIPv6のみで運用されることは推奨されていないのです。

RFC 3901には、「IPv4で名前解決ができない状況の発生を避けるように推奨しているのは、

---

†25 RFC 8482：J. Abley, O. Gudmundsson, M. Majkowski, E. Hunt, “Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY”, 2019年1月

†26 RFC 3901：A. Durand, J. Ihren, “DNS IPv6 Transport Operational Guidelines”, 2004年9月

CC BY NC SA

名前空間のフラグメント化を避けるためである」と書かれています。DNSによる名前空間はルートからの階層構造になっているので、その階層構造の一部が特定のバージョンのインターネットプロトコルにおいてのみ確認可能であるのは、名前空間のフラグメント化にほかならないからです。IPv4インターネットのみだった時代には名前空間がフラグメント化されることがなかった一方で、IPv6のみによる名前空間はフラグメント化を発生させてしまうという主張です。

16.8 廃止された仕様

IPv6のためにDNSに対して施された拡張仕様のいくつかは、その後廃止されました。以下、廃止された旧仕様である`ip6.int`とA6リソースレコードを紹介します。

16.8.1 逆引きのための`ip6.int`の廃止

1995年に発行されたIPv6のためのDNS拡張の仕様を示したRFC 1886では、IPv6アドレスの逆引きのために利用するドメインとして、`ip6.int`が記述されていました。しかし、2001年に発行されたRFC 3152によって、`ip6.int`が果たしていた役割が、2000年に発行されたRFC 2874で定義された`ip6.arpa`へと委任されました。さらに、その後、2003年に発行されたRFC 3596によって、逆引きで`ip6.int`を利用するとしていたRFC 1886が上書き廃止され、逆引きとして`ip6.arpa`を利用することが正規の仕様になりました。そして、2005年に発行されたRFC 4159によって`ip6.int`の運用も廃止されました。

`ip6.int`が廃止された背景として、RFC 1591にあるように、`.int`というトップレベルドメインが国際機関と国際的なデータベースのために予約されていた点が挙げられます。しかし、2000年にIABが国際機関と国際的なデータベースは別々の名前のもとで運用すべきであるという見解を表明し、国際的なデータベースとしての役割は`.int`から`.arpa`へと移行されました。2001年には`.arpa`トップレベルドメインの用途に関してRFC 3172が発行されています。その中では、`in-addr.arpa`、`ip6.arpa`、`e164.arpa`の3つが記載されています。

本書執筆現在、`.int`トップレベルドメインは国際条約に基づく国際機関のみが利用可能です[†27]。`.int`を利用している国際機関としては、たとえば国連（`un.int`）が挙げられます。

`ip6.arpa`が`ip6.int`から置き換えられる過程では、それまでルートDNSサーバで管理されていた`.arpa`トップレベルドメインをルートDNSサーバから切り離すことも検討されていましたが、`.arpa`トップレベルドメインは13個あるルートDNSサーバのうち`j`を除く12個によって運用されています[†28]。`.arpa`トップレベルドメインをルートDNSサーバから切り離すことに関しては、RFC 2870の2.5で紹介されています[†29]。

†27 https://www.iana.org/domains/int/policy

†28 Root Server System Advisory Committee (RSSAC), "Service Expectations of Root Servers", November 2014, https://www.icann.org/en/system/files/files/rssac-001-root-service-expectations-04dec15-en.pdf の「2. Service Provided by Root Servers」参照

†29 RFC 2870はRFC 7720によって上書き廃止されています。

CC BY NC SA

16.8.2 廃止されたA6リソースレコード

IPv6アドレスを示すためのリソースレコードとして、AAAAレコードのほかにA6レコードというものもありました。Draft StandardのRFC 1886でAAAAレコードが定義されたのは1995年でしたが、2000年にProposed StandardのRFC 2874でA6レコードが定義されました。当初Proposed StandardであったA6レコードは、2002年にRFC 3363によってExperimentalへと変わりました。同時期に発行されたRFC 3364では、AAAAとA6を比較しつつ、A6が持つ多くの問題点がまとめられています。

つまり、A6レコードは2000年に標準として定義され、その2年後に実験的というステータスに降格しました。その後、2012年にRFC 6563でA6レコードが廃止されるまでは、IPv6アドレスを示すためのリソースレコードが2種類存在していたことになります。

A6リソースレコードは、IPv6によるネットワークリナンバリングやマルチホームを扱いやすくするために、チェイニング（chaining）と呼ばれる方法でIPv6アドレスを分割して管理できるリソースレコードでした。しかし、A6レコードの解決のためにDNSサーバに対して多くの問い合わせが発生することや、処理が複雑になってしまう可能性があること、管理が複雑になってしまう可能性、セキュリティリスクが高くなる可能性などが指摘されていました。さらに、1つの目的のために2つの異なるリソースレコードが存在したときに、同じ名前解決でリソースレコードによって結果が異なるときにどうするかという問題もありました。そのため最終的にA6は廃止され、本書執筆現在ではAAAAのみがIPv6での名前解決で利用されています。

本書執筆現在は`ip6.arpa`がIPv6アドレスの逆引きで利用されていますが、`ip6.arpa`の利用はA6レコードを最初に定義したRFC 2874で定義されています。RFC 2874の内容のうち、A6レコードが廃止された一方で、`ip6.arpa`については`ip6.int`が廃止されるという背景で生き残ったのは興味深いところです。

第17章

DNSによるデュアルスタック環境の実現と運用

2つの異なるインターネットプロトコルであるIPv4とIPv6の間には直接的な互換性はありません。しかし実際には、同一の物理回線を利用し、IPv4とIPv6の**デュアルスタック**環境で運用されることが一般的です。直接的な互換性がないIPv6インターネットとIPv4インターネットを、一般ユーザが1つのインターネットとして利用できるのは、DNSのおかげです。

DNSによってIPv4とIPv6のデュアルスタック環境が実現できるとはいえ、IPv4という単一のインターネットプロトコルで運用されていたネットワークがデュアルスタック環境になることはかなり大きな変化であり、運用に関する教育やノウハウの蓄積もこれからです。IPv4のネットワーク設定とIPv6のネットワーク設定の勘違いなどによる障害の発生や、どちらか片方だけのプロトコルで障害が発生している場合をどうやって切り分けるかといった課題もあります。本章では、そうした問題や課題についても紹介します。

17.1 デュアルスタック環境の実現

インターネットは、単一の名前空間を構成するように設計されています（RFC 2826[†1]）。しかし、同一の名前に対応するIPv4アドレスとIPv6アドレスをそれぞれ解決できるだけでは、単一の名前空間を維持しながらのIPv4とIPv6のデュアルスタック環境の実現にとって不十分でした。そのため、IPのバージョンに依存して名前空間が別物にならないようにするため、DNSによって管理される名前の内容と、DNSサーバにアクセスするときにパケットを配送（トランスポート）するネットワーク層プロトコルのバージョンを、それぞれ独立にするという設計が選択されました。具体的には、IPv4を使ってIPv6に関連する名前解決が行え、逆に、IPv6を使ってIPv4に関連する名前解決もできるような設計になりました。

17.1.1 DNSサーバへのトランスポートにIPv6とIPv4を利用可能にする

IPv4およびIPv6と、DNSメッセージのResource Recordに関する組み合わせとしては、表17.1に示す4パターンが考えられます。

†1 RFC 2826：Internet Architecture Board, “IAB Technical Comment on the Unique DNS Root”, 2000年5月

cc by nc sa

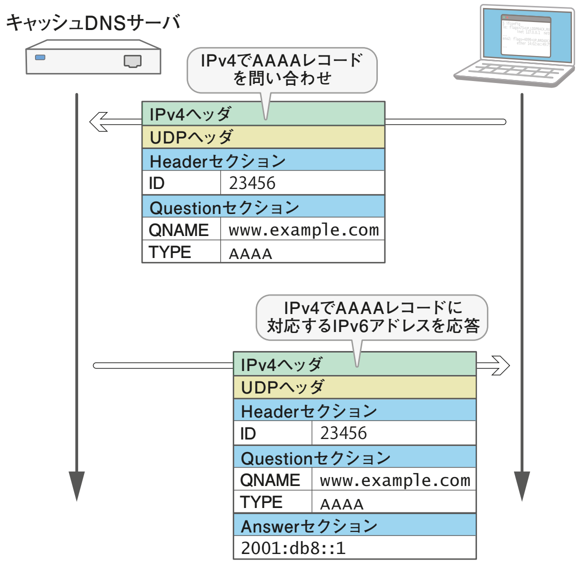
▶ 表17.1　IPv4/IPv6とDNSメッセージにおけるResource Recordの組み合わせ

トランスポート	Resource Record
IPv4	A
IPv4	AAAA
IPv6	A
IPv6	AAAA

もし、AAAAの検索がIPv6でのみ可能という設計だったとすると、ある名前解決に関与するすべてのDNSサーバがIPv6対応している必要が生じます。逆に言うと、IPv6のみに対応した権威DNSサーバだけで運用してしまうと、その権威DNSサーバが扱う名前空間に関してはIPv4でまったく名前解決ができなくなってしまいます。

そのため、RFC 3901[†2]では、名前管理を行っている組織が運営する権威DNSサーバのうちの少なくとも1つはIPv4でもアクセス可能にすることを推奨しています。この推奨は、IPv6に関するDNSの運用上の問題を扱ったRFC 4472[†3]にも引き継がれています。

したがって、たとえば図17.1のように、IPv6を示すAAAAレコードの問い合わせに際してDNSサーバへのトランスポートにIPv4が利用できます。



▶ 図17.1　IPv4でAAAAレコードを問い合わせる場合

逆に、IPv6をDNSサーバへのトランスポートとして利用し、IPv4を示すAレコードを問い合わせることもできます。

ルートDNSサーバについては、2008年2月4日に13系統あるルートDNSサーバのうちの6系統にIPv6アドレスが設定され、IPv6トランスポートによるルートDNSサーバへの問い合わ

†2　RFC 3901：A. Durand, J. Ihren, "DNS IPv6 Transport Operational Guidelines", 2004年9月

†3　RFC 4472：A. Durand, J. Ihren, P. Savola, "Operational Considerations and Issues with IPv6 DNS", 2006年4月

cc BY NC SA

せが可能になりました。つまり、2008年2月以前は、IPv6トランスポートでルートDNSサーバへの問い合わせができなかったということです。もし仮にIPv6トランスポートでしかAAAAレコードの問い合わせができない仕様だったなら、2008年2月以前にはAAAAレコードを利用した名前解決がすべて不可能で、したがってIPv6アドレスの名前解決ができなかったことでしょう（もちろん実際にはIPv6でインターネットの名前解決は可能でした）。

17.1.2 IPv4アドレスとIPv6アドレスは同時に問い合わせできない

名前に対するIPv4アドレス（Aレコード）の問い合わせとIPv6アドレス（AAAAレコード）の問い合わせを別々に行う必要があるので、DNSサーバに対する問い合わせは2回発生します。

DNSサーバに対してIPv4アドレスとIPv6アドレスの問い合わせを同時に送ることはできません。これは、DNSでは1回の問い合わせに対して利用できるメッセージは1つのみだからです。複数の問い合わせを1つのDNSメッセージとして送信することは、基本的にはできません。

DNSの仕様上はIPv4とIPv6を同時に問い合わせ可能？

実は、DNSサーバへのIPv4アドレスとIPv6アドレスの問い合わせを別々に行わなければならないのは、「事実上そうなっている」という話です。「仕様上そうなっている」とは言い切れません。

DNSに関するRFCは、RFC 1034[†4]とRFC 1035[†5]を基本としています。RFC 1034でDNSのコンセプトが解説され、RFC 1035で仕様が記載されている格好です。これらのRFCは、1987年に発行された歴史あるものですが、本書執筆時点においてRFC 1034とRFC 1035を上書きして廃止するRFCは存在しないので、DNSに関してはRFC 1034とRFC 1035がベースになっています。

RFC 1035でQuestionのフォーマットを規定しているSection 4.1.2には、“The section contains QDCOUNT (usually 1) entries”と書かれています。この記述があることから、仕様上は「DNSサーバへのIPv4アドレスとIPv6アドレスの問い合わせを同時にできる」と読むこともできてしまうのです。

この記述の何が問題かを読み解いてみましょう。そもそも、QDCOUNTのQDが何を意味するか明確に示されていないのが困りものです。QはQuestionだと推測されますが、Dが何なのかは謎です。そして、そのQDCOUNTが「普段は1」（“usually 1”）と記述されているわけです。

Questionセクションに含まれるエントリが「普段は1」ということは、普段とは違う場合があると考えられるので、RFCの仕様上は複数のQuestionを同時に送信することが明確に禁止されていないとも読めます。つまり、RFC 1035を読む限りは、QDCOUNTを2にしたうえでQuestionセクションとしてIPv4用のAレコードとIPv6用のAAAAレコードを同時に送信すれば、IPv4アドレスとIPv6アドレスに関する問い合わせを同時にできなくはない、ともとれます。RFC 1035の発行は1987年で、当時はいまほど厳密な記述でないとはいえ、これまで上書きされずにそのまま仕様として残り、その仕様上はIPv4アドレスとIPv6アドレスに関するDNSでの問い合わせが同時に可能という状態なのです。

CC BY NC SA

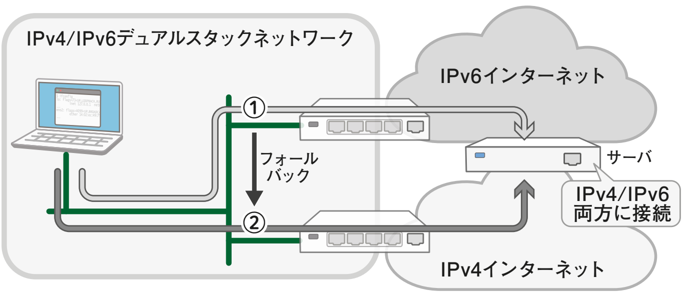
ただし、現時点ではQDCOUNTが1ではない値を受け付ける実装がありません。したがって、複数の問い合わせを同時に含むDNSメッセージを受け入れる実装は存在せず、単一のDNSメッセージで行える問い合わせは事実上1つだけです。

昔のRFCに謎の仕様が残されていることは少なくありませんが、DNSというインターネットの根幹的な仕組みにおいてさえ修正されずに残っている謎の仕様が存在し、それに基づいて世界中で運用されているのが現在のインターネットの実状です。

17.2　IPv6からIPv4へのフォールバック

DNSは、ユーザが実行するアプリケーションがIPv4を使って通信を行うのか、それともIPv6を利用するのかを判断する際に、交通整理役として大きな役割を果たします。ただ、この「実は2つだけど交通整理役のおかげで1つに見える」という状況によって問題が引き起こされる場合もあります。

通信を開始する側のホストがIPv4とIPv6の両方を利用していて、DNSによる名前解決で通信相手のFQDNに対応するIPv4とIPv6両方のアドレスが取得できたとすると、通信を開始する側のホストはまずIPv6による通信を試みます。これは、IPv6とIPv4の両方のIPアドレスが存在する宛先と通信する場合に、「まず最初にIPv6を試してみて、それでダメであればIPv4を試す」というのが一般的な方法だからです（図17.2）。IPv6での通信が成功すれば問題がないのですが、もしIPv6による通信ができない状況だと、IPv4での通信へのフォールバックが起こることになります。



▶ 図17.2　IPv6からIPv4へのフォールバック

現時点のIPv4とIPv6デュアルスタック環境では、このようなIPv6からIPv4へのフォールバックという現象が発生する可能性があります。DNSへの問い合わせそのものはIPv4とIPv6のどちらを利用することも可能なので、権威DNSサーバにIPv4とIPv6の両方のアドレスが登録されていれば、IPv4しか利用できない環境であってもFQDNに対応するIPv6アドレスを取得できてしまうからです。これは、互換性がないIPv4とIPv6とでDNSを利用して名前解決を行う部分は共通であることから発生する現象であり、IPv4とIPv6とで同じ名前空間を共有するという設計に起因する問題だといえます。

†4　RFC 1034：P.V. Mockapetris, “Domain names - concepts and facilities”, 1987年11月
†5　RFC 1035：P.V. Mockapetris, “Domain names - implementation and specification”, 1987年11月

IPv6からIPv4へのフォールバックの問題は、通信開始までの遅延です。まずIPv6で試してみて、それがダメならばIPv4で通信することになるので、従来のようにIPv4だけで通信する場合より通信開始までに時間がかかります。IPv6での接続が失敗してIPv4での接続が確立するまで、数十秒も余分な時間を要する可能性すらあります。

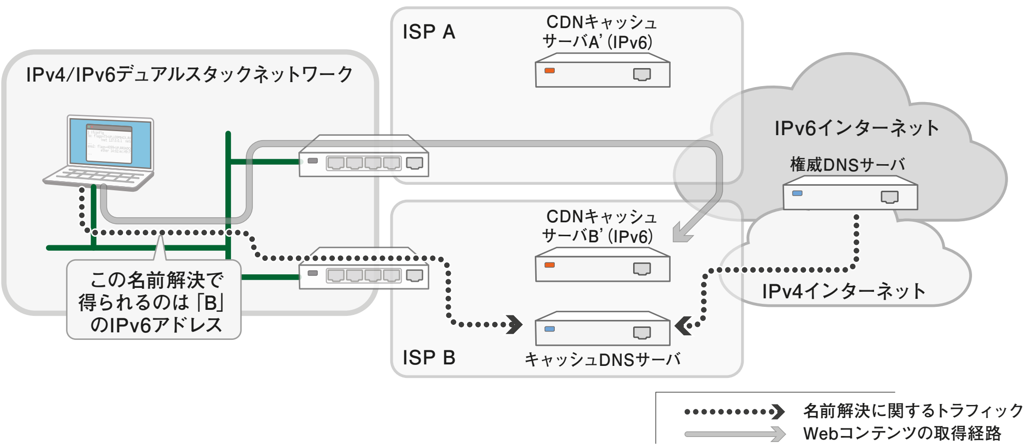
さらに大きな問題は、IPv6での接続に失敗してしまうと通信そのものを諦めてしまうアプリケーションが存在する点です。たとえば、メールソフトやWebブラウザのバージョンによっては、IPv6での通信が失敗したあとにIPv4での通信を試みないものもあります。このような問題は、IPv4アドレス在庫枯渇が現実のものとなってIPv6に関する話題が増えたこともあって徐々に修正されつつありますが、古いソフトウェアを使い続けている場合には注意が必要です。

それ以外にも、IPv6からIPv4へのフォールバックには、実際の通信で使われているのがIPv4かIPv6かがわからなくなって障害発生時に問題の切り分けに時間がかかる可能性があるなど、副作用ともいえる影響もあります。

17.3　キャッシュDNSサーバとCDNに関する問題

Webトラフィックなどの負荷分散を目的として利用されるCDN（Content Delivery Network）で、キャッシュDNSサーバのIPアドレスが利用される場合があります。IPv4とIPv6のデュアルスタック環境において、サーバとクライアント間のパケットが通過する経路がIPv4とIPv6とで異なると、どのキャッシュDNSサーバを利用したかによって通信効率が大きく変わってしまう可能性があります。これは、DNSによる問題というよりも、キャッシュDNSサーバのIPアドレスを利用した負荷分散を行うCDNとの相性問題といえます。既存のCDNの構築手法がある意味で無理矢理な力技なので、技術的に何か問題があるというよりは、既存の運用では効率化が十分にできないだけという解釈も可能です。

この問題が発生するのは、権威DNSサーバが問い合わせを行ったキャッシュDNSサーバに応じて回答するIPアドレスを変更することでCDNを実現している環境です。図17.3に問題が発生している状況を示します。



▶図17.3　デュアルスタック環境におけるCDN

図17.3では、IPv6はISP Aによって提供され、IPv4はISP Bによって提供されています。ユーザが利用しているPCに設定されているキャッシュDNSサーバは、IPv4を利用しているISP Bのものを利用しています。

このような環境で、CDNによって負荷分散が行われているコンテンツをユーザが取得するときの流れを見てみましょう。まず、ユーザがISP BにあるキャッシュDNSサーバを利用して名前解決を行います。このとき、ユーザはIPv4とIPv6の接続性を両方持っているので、AAAAレコードの問い合わせを先に行っているものとします。AAAAレコードの問い合わせに使われるキャッシュDNSサーバまでのトランスポートは、この環境では当然IPv4になります。

ISP BにあるキャッシュDNSサーバは、権威DNSサーバにAAAAレコードの問い合わせを行い、回答を得ます。ISP Bは、IPv4とIPv6の両方で運用されており、IPv6で運用されたCDNキャッシュB'がISP B内にあるものとします。このような環境下で、権威DNSサーバからキャッシュDNSサーバへ返答されるのは、問い合わせを行ったキャッシュDNSサーバの最寄りにあるコンテンツキャッシュであるCDNキャッシュB'のアドレスです。

キャッシュDNSサーバからの名前解決結果を受け取ったユーザは、CDNキャッシュB'からコンテンツを取得します。このとき、コンテンツの取得に利用されるのはIPv6なので、ユーザはISP Aを経由してCDNキャッシュB'にあるコンテンツを取得します。

しかし、ISP AにもCDNキャッシュA'というコンテンツキャッシュがある場合、これではユーザは遠回りをしてコンテンツを取得していることになってしまいます。このような遠回りが発生するのは、CDNによる負荷分散機能の一部である権威DNSサーバが「ユーザはISP Bにいる」と認識しているために発生しています。

ISP AのキャッシュDNSサーバを利用すれば、ユーザはCDNキャッシュA'からIPv6トランスポートを利用してコンテンツを取得できるようになりますが、今度は逆にIPv4でのCDNキャッシュコンテンツ取得が遠回りになってしまいます。

IPv4のAレコードを取得するときと、IPv6のAAAAレコードを取得するときに利用するキャッシュDNSサーバを切り替えられればよいのですが、そういった使い分けが行われることは稀なので、このような問題が発生します。

17.4　デュアルスタック環境におけるSRVの利用

SRVは、「Service」を意味するResource Recordで、インターネット上で提供されているサービスの場所を探すために使われます。各種サービスを利用したいクライアントは、権威DNSサーバに登録されているSRVを通じて、そのサービスに必要なサーバを発見できます。SRVはRFC 2782[†6]で規定されています。

たとえば、example.comというドメインで運用されているUDPのSTUNサーバを探すとき、クライアントは次のようなフォーマットでSRVの問い合わせを行います（STUNについては第20章で説明します）。

```
_stun._udp.example.com
```

†6　RFC 2782：A. Gulbrandsen, P. Vixie, L. Esibov, “A DNS RR for specifying the location of services (DNS SRV)”, 2000年2月

CC BY NC SA

これに対して得られるのは、IPアドレスではなく、そのサービスのFQDNとポート番号などの情報です。クライアントは、IPv4もしくはIPv6アドレスを取得するために、得られたFQDNを利用してAもしくはAAAAのDNS問い合わせを改めて行います。

ここで問題になるのが、単一のFQDNに対してIPv4とIPv6とでポート番号を変えてサービスするといった運用ができないことです。たとえば、同じFQDNで、IPv4ではTCPポート番号30080、IPv6ではTCPポート番号80でサービスを提供するといったことはできません。そのため、IPv4とIPv6とでサービスを分けるには、IPv4用とIPv6用で異なるFQDNを用意してSRVレコードを別々に設定する必要があります。

17.5 IPv6 DNSホワイトリスティングとブラックリスティング

特定のサーバに対してIPv4で通信を行うのか、それともIPv6で通信を行うのかは、権威DNSサーバの設定に大きく依存します。権威DNSサーバにIPv6用のAAAAレコードがあると、IPv6を利用可能なクライアントはIPv6で通信を始めようとするからです。しかし、IPv6対応ができていない組織に対してAAAAレコードを含むDNS応答をしてしまうと、前述したIPv6からIPv4へのフォールバックが発生したり、そもそも通信に失敗したりします。大量のトラフィックを日々処理しているWebコンテンツ事業者が、ある日突然IPv6対応をすると、それが原因で多くのユーザがサービスを円滑に利用できなくなる可能性があるのです。そのような状況を招かないよう、IPv6対応を徐々に進めたいという要望から、IPv6 DNSホワイトリスティングという手法が使われることがあります。

IPv6 DNSホワイトリスティングは、「IPv6対応が行われているとわかっている組織に対してだけAAAAレコードを返答する」というものです。IPv6 DNSホワイトリスティングとは逆に、「AAAAを返すと障害の原因となりそうな相手にはAAAAを返さない」というIPv6 DNSブラックリスティングという方式もあります。

IPv6ホワイトリスティングとブラックリスティングは、エンドユーザに対してIPv6でのコンテンツ提供をするうえでの注意点をまとめたRFC 6589[†7]で詳しく紹介されています[†8]。

17.5.1 IPv6 DNSホワイトリスティング

IPv6 DNSホワイトリスティングは、実験的にIPv6に取り組みたいと考えるコンテンツ事業者によって使われることが多い方式です。この方式に従って運営されている権威DNSサーバは、ホワイトリストに登録されたキャッシュDNSサーバからのAAAA問い合わせだけに答えます。

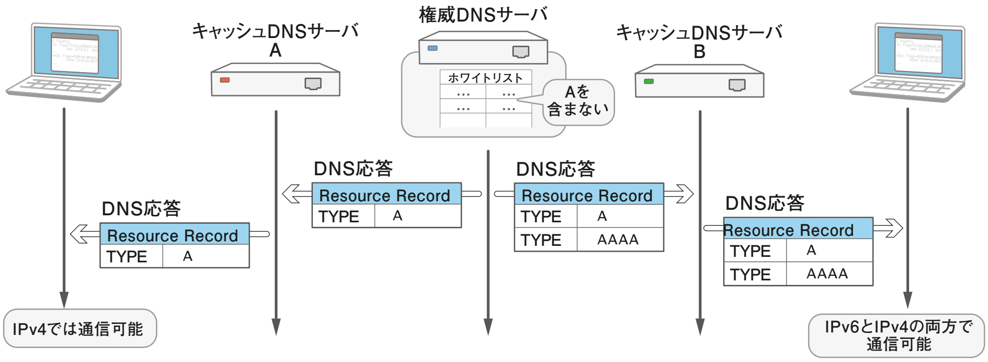
図17.4の左側が、ホワイトリストに掲載されていないキャッシュDNSサーバからの名前解決問い合わせへの対応です。右側が、ホワイトリストに掲載されているキャッシュDNSサーバからの問い合わせへの対応を示しています。ホワイトリストに掲載されているキャッシュ

†7 RFC 6589：J. Livingood, “Considerations for Transitioning Content to IPv6”, 2012年4月

†8 本書執筆時点ではInternet Draftですが、“IPv6 Guidance for Internet Content and Application Service Providers” https://tools.ietf.org/id/draft-carpenter-v6ops-icp-guidance-03.html も非常に参考になります。

CC BY NC SA

DNSサーバに対しては、AレコードとAAAAレコードの両方にIPアドレス情報を返しますが、ホワイトリストに掲載されていないキャッシュDNSサーバに対してはAレコードのみIPアドレス情報を返します。

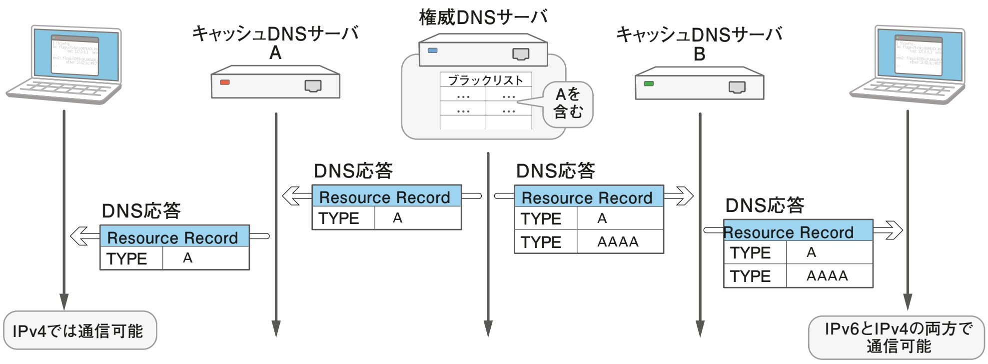


▶ 図17.4　IPv6 DNSホワイトリスティング

Googleは、2008年3月から2012年6月のWorld IPv6 Launchまで、IPv6 DNSホワイトリスティングによるAAAAレコード提供を行っていました。これは、Google over IPv6と呼ばれるプログラムに申し込んだ組織だけがIPv6でGoogleのコンテンツを取得できるというものでした。2010年1月以降は、Google over IPv6に参加している組織であればYouTube動画をIPv6で閲覧できるようになっていました。

17.5.2　IPv6 DNSブラックリスティング

IPv6 DNSブラックリスティングは、ブラックリストに掲載されたキャッシュDNSサーバにのみAAAAレコードを返さないというものです。デフォルト設定としてはAAAAレコードに返答し、問題を発生させる可能性が高いと思われるキャッシュDNSサーバに対してはAレコードのみを返します。



▶ 図17.5　AAAAブラックリスティング

(CC BY NC SA)

ネットワーク技術においてブラックリストという用語は通信を拒否する対象の一覧として使われてきました[†9]。しかしIPv6 DNSブラックリスティングはまったく通信できなくするわけではなく、基本的にIPv4での通信は可能であり続けます。そのため、実際のインターネットではブラックリストに掲載されたことに気がつかない可能性もあります。とはいえ、長期的視点で見ると、ブラックリストに掲載されたキャッシュDNSサーバを利用するネットワークにおいてIPv6の普及が阻害されるなどの影響はありえます。

Googleは、2012月6月6日のWorld IPv6 Launch以降、Google over IPv6のようなプログラムへ申請のあった組織にのみAAAAレコードを返すIPv6 DNSホワイトリスティングではなく、問題を発生させる可能性がある組織に対してAAAAレコードを返さないという方式に切り替えています。つまり、「デフォルトではAAAAレコードを返さず、申請があった組織に対してのみAAAAレコードを提供する」という状態から、「AAAAレコードを返さないほうがよいとGoogleが判断した組織に対してのみAAAAレコードを返さない」という方式に変わりました。Googleが作成したブラックリストはFacebookなど他のコンテンツ事業者も利用しています。

日本では、Googleのようなコンテンツ事業者が各自の権威DNSサーバで運営するブラックリストとは別に、ISPにあるキャッシュDNSサーバでAAAAフィルタが実施されている場合があります。これは、NTTフレッツ網における閉域IPv6問題に対処するためです。キャッシュDNSサーバでのAAAAフィルタに関してはA.1.4項で説明します。

17.6 アプリケーションのIPv6対応とデュアルスタック環境

ネットワークやOSのIPv6対応と、各機器にインストールされているアプリケーションのIPv6対応は、独立した問題です。ネットワークとOSがIPv6に対応してIPv4とIPv6のデュアルスタック環境が実現されたとしても、アプリケーションがIPv6対応されなければユーザがIPv6を利用することはないのです。アプリケーション実装者がIPv6対応コードをアプリケーションに追加するまでは、そのアプリケーションがデュアルスタック環境でIPv6を使うことはありません。アプリケーションのIPv6対応に関しては、RFC 4038[†10]で概説されています。

アプリケーションに関してIPv4とIPv6のデュアルスタック環境で注意が必要なのは、ソフトウェアのバージョンによって挙動が異なる可能性です。同じアプリケーションであっても複数の異なるバージョンが同時に利用されることがあり、バージョンによってIPv6対応状況が異なる場合もあります。

IPv6には、IPv4-Mapped IPv6アドレスという仕組みがあり、IPv6を利用してIPv4アドレスと通信しているかのような挙動を実現することができます。一見便利な仕組みですが、デュアルスタック環境では、IPv4での通信のようにIPv6を偽装することでフィルタをすり抜けてしまうといったセキュリティ上の懸念があります。IPv4-Mapped IPv6アドレスを使うつもりがない場合には、IPv6ソケットに対して明示的に`IPV6_V6ONLY`オプションを設定しておくことをお勧めします。IPv4-Mapped IPv6アドレスに関しては3.9節を参照してください。IPv6対

---

†9 現在は「ブラックリスト」「ホワイトリスト」のような表現より「拒否リスト（denylist）」「許可リスト（allowlist）」という表現もよく使われますが、本書ではRFC 6589の用語に合わせています。

†10 RFC 4038：M-K. Shin, Y-G. Hong, J. Hagino, P. Savola, E. M. Castro, “Application Aspects of IPv6 Transition”, 2005年3月

cc BY NC SA

応プログラミングに関しては第15章を参照してください。

なお、アプリケーションのIPv6対応は、必ずしもアプリケーションにおけるIPv4とIPv6のデュアルスタック化を意味するわけではありません。実際、さまざまなネットワークコマンドでは、IPv4用とIPv6用とが別のアプリケーションになっています。たとえば、Unix系システムで利用されることが多い`ping`コマンドは、IPv4の場合にはICMP Echo Requestを送信するように実装されています。この`ping`コマンドのIPv6版は、`ping6`コマンドとして、IPv4とは別のアプリケーションになっています。

WebのIPv6対応は他人事ではない

できるだけ高速にサイトを表示することが重要とされるWebコンテンツ事業者にとっては、IPv6からIPv4へのフォールバックによって表示速度が遅くなったり、そもそも表示されなくなったりするのは、なんとしても避けたい状況です。そのような事情から、WebサイトのIPv6化について社内の承認を得るのが難しいという話も少なくありません。

しかし、Webにおいては、「自分のサイトで直接対応しない技術であっても間接的に影響がある」という場合がよくあります。IPv6対応も例外ではありません。「自分のサイトをどのようにしてIPv6対応するか」という課題とは別に、外部Webサイトの画像表示、外部Webサイトへのリンク、外部WebサイトAPIの利用などで、IPv6に関連した対応が発生する場合があります。たとえば、画像を提供しているアフィリエイト事業者がIPv6に対応すると、「IPv6で取得可能な画像」がWebサイト内で利用されるようになります。IPv6対応された画像が表示されなかったりリンク先に移動したりできなければ、自分のWebサイトがIPv6対応を行っていなくても、IPv6対応の影響を受けてしまっていることになります。

「自分はIPv6対応をするつもりがないので関係ない」と言い切れないところが、デュアルスタック環境におけるIPv6対応の難しい点のひとつかもしれません。

CC BY NC SA

第IV部

IPv4はどうなるのか

本書は、IPv4アドレスの中央在庫が枯渇してから何年も経過した時点で執筆しています。にもかかわらず、IPv4によるインターネットはいまだに運用され続けています。IPv4アドレスの在庫が枯渇しても、IPv4インターネットは継続しているのです。

「IPv4アドレス在庫は枯渇してない」と考える人も少なくありません。その背景には、「IPv4アドレス在庫」には複数の種類があり、地域ごとに「IPv4アドレス在庫の枯渇」の実態が異なることがあります。必然的に「IPv4アドレス在庫が枯渇しました」というニュースをたびたび耳にすることになり、狼少年の寓話のような状況になっているのでしょう。

本書の主題であるIPv6は、IPv4アドレスの枯渇対策として説明されることがよくあります。だとすれば、IPv4アドレスが枯渇しても利用し続けていられるいま、IPv6への移行にはどのような意味があるのでしょうか。

本書の視点のひとつは、IPv4アドレス在庫枯渇への直接的な対策と、IPv6対応を、個別の話題として捉えることです。この部では、まずIPv4アドレス在庫枯渇とはそもそも何であるか、それによって何が起きるのかを説明します。そして、この問題に対する短期的解決策で利用される次のような技術について説明します。

- IPv4 NATとプライベートIPv4アドレス
- CGN
- STUN、TURN

cc BY NC SA

[書籍版では空白のページです]

BY NC SA

第18章

IPv4アドレス在庫枯渇とその解決策

IPv6はIPv4アドレス在庫枯渇問題に対する長期的解決策として作られました。「IPv4アドレスが枯渇したから、その対策としてIPv6への移行を急ぐべき」というのは、マクロな視点で見た場合は確かに解決策です。インターネット全体や、国としてという視点で見たときには、IPv6への移行がIPv4アドレス在庫枯渇への解決策といえます。

しかし本書では、IPv6を「IPv4アドレス在庫枯渇問題に対する解決策」としては説明していません。これは、IPv4とIPv6には直接的な互換性がないからです。現在のインターネットは、IPv6を利用可能な環境は急激に増えつつあるものの、IPv4をまったく利用できなくても困らない状況とは言い難い状況です。たくさんのユーザがIPv4を利用している状況で、IPv6で何かをしても、IPv4ユーザが直接それを利用できるわけではありません。IPv4によるサービス提供は、引き続き重要であり続けます。IPv6への移行は、IPv4アドレス在庫枯渇に対する直接的な解決策とは必ずしも言い切れないのです。

IPv4アドレス在庫枯渇には、IPv6への移行ではなく、IPv4上での対策が必要です。言い換えれば、IPv4アドレス在庫枯渇に対する直接的な対策と、IPv6対応は、個別に考えるべき課題です。

この章では、IPv4アドレス在庫枯渇に関して詳しく紹介するとともに、限られたIPv4アドレス資源を有効活用するための技術を紹介します。

18.1 IPv4アドレスの在庫枯渇はどのような問題なのか

IPv4アドレスの在庫枯渇に関しては、「アナログ放送の停波と地デジへの移行」や「原油の枯渇」に似た事態であるという認識に基づく説明が多く見られます。しかし、これらの事態は、IPv4アドレスの枯渇について説明するアナロジーとしては不適当です。むしろ、これらの事態とIPv4アドレスの枯渇との相違点を考えてみることで、IPv4アドレスの枯渇とは何であるかが見えてきます。

まず、アナログ放送の停波との相違点を考えてみましょう。日本では、2011年7月にテレビ用のアナログ放送が停波し、地デジ放送へと移行しました。その際に起きたのは、ある日突然アナログ放送が視聴できなくなるという事態でした。一方、IPv4アドレスの在庫が枯渇す

るという問題では、ある日突然IPv4アドレスが使えなくなるわけではありません。それまで使っていたIPv4アドレスはそのまま使い続けられるという意味で、IPv4アドレスの枯渇はアナログ放送の停波とは大きく異なります。

原油枯渇問題も、IPv4アドレス在庫枯渇のアナロジーとしては、やはり適切ではありません。原油は使うとなくなってしまいますが、IPv4アドレスは使い続けるものです。利用することによって「消費」されて減るものではありません。

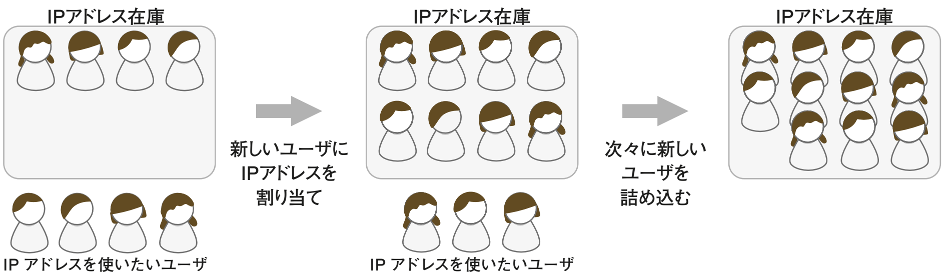
筆者がIPv4アドレスの在庫枯渇問題を説明する際によく使うアナロジーは、「土地」や「相撲の親方株」です。土地は、まだ誰も所有していない部分が大量にある間は、新しく割り当てが可能です。すべての土地の割り当てが終了してしまうと、限られた土地を皆で共有しながら使わなくてはなりません。相撲の親方株（年寄名跡）は、日本相撲協会によって定員が105と決まっており[†1]、その限られた資源の権利を譲渡しながら運用が行われています。土地や相撲の親方株に共通するのは、「有限な資源であり、上限に達することがある」という点です。上限に達した場合、それ以上は新規に割り当てられないので、既存の資源を共有したり譲渡したりしながら活用していかざるを得ません。資源の利用者もしくは所有者が誰であるかを管理している組織があるという点も、IPv4アドレスに共通しています。

18.2　IPv4アドレスの在庫枯渇の影響

IPv4アドレスの在庫が枯渇すると何が起こるのでしょうか。ものすごく単純に言えば、IPv4アドレスが枯渇することで、それ以上IPv4によるインターネットが拡大しにくくなります。

IPv4アドレス在庫枯渇が発生する前のIPv4インターネットでは、図18.1の中央の図のように、新規ユーザが参加したいと言ったときに新しいIPv4アドレスを割り当てて対処してきました。IPv4アドレスの在庫があったので、ユーザが増えるぶんだけ新しくIPv4アドレスの割り当てが可能な状態だったのです。IPv4アドレス在庫枯渇が発生する前のIPv4インターネットは、IPv4アドレス在庫という制約に制限されることなく、どんどん拡大していきました。

IPv4アドレスの在庫が枯渇してくると、限られたIPv4アドレスの範囲内で新規ユーザに対処しなければならなくなります。限られた空間に次々に新しいユーザを詰め込めば、図18.1の右側の図のように、一人あたりのIPv4アドレス数が減ってしまいます。



▶ 図18.1　IPv4アドレス在庫枯渇前の状態

†1　現役時代の功績が著しかった力士に対して例外的に一代に限り特別に認める一代年寄は、105に含まれません。

CC BY NC SA

IPv4 アドレス在庫枯渇によって発生する状態は、この「詰め込み度合い」の上昇です。IPv4 インターネットへの参加者が増えなければ、一人あたりが利用できる IPv4 アドレス数は減らないので、IPv4 アドレス在庫が枯渇しても大きな問題にはなりにくいともいえます。現実には IPv4 インターネットの成長が続いているので、IPv4 アドレスの在庫枯渇によってさまざまな影響が起こりえます。

18.3 IP アドレス管理の階層構造から見た IPv4 アドレスの在庫枯渇

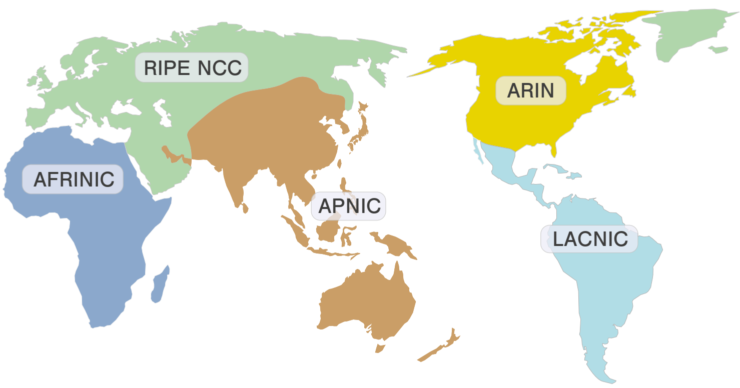
ここまでの説明で「IPv4 アドレスの在庫」と呼んでいたものは、どのように管理されているのでしょうか。実は世界にはいくつかの「IPv4 アドレス在庫」があり、それらが階層的な構造をなしています。ここでは、この IP アドレス管理の階層構造について説明し、その観点から IPv4 アドレスの在庫枯渇問題を改めて整理します。

18.3.1 IP アドレス管理の階層構造

インターネットの IP アドレスは一意であることが求められます。そのため、誰がどこでどのような IP アドレスを使うかに関しては、全世界で一元的に管理されています。IP アドレスの一元管理は、IANA（Internet Assigned Numbers Authority）を頂点とする階層構造で行われています。

IANA は、すべての AS 番号や IP アドレスを管理していますが、IANA が自ら直接ユーザに割り当てる業務はしません。IANA が管理する AS 番号や IP アドレスは、まず世界 5 地域を代表する RIR（Regional Internet Registry、地域インターネットレジストリ）へと割り振られます。

- AFRINIC（アフリカ）
- APNIC（アジア太平洋地域）
- ARIN（北米地域）
- LACNIC（ラテンアメリカおよびカリブ海地域）
- RIPE NCC（ヨーロッパ、中東、中央アジア）

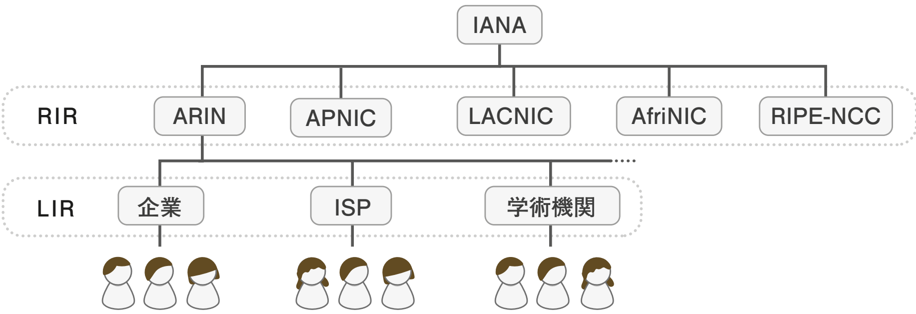


▶ 図 18.2　RIR

CC BY NC SA

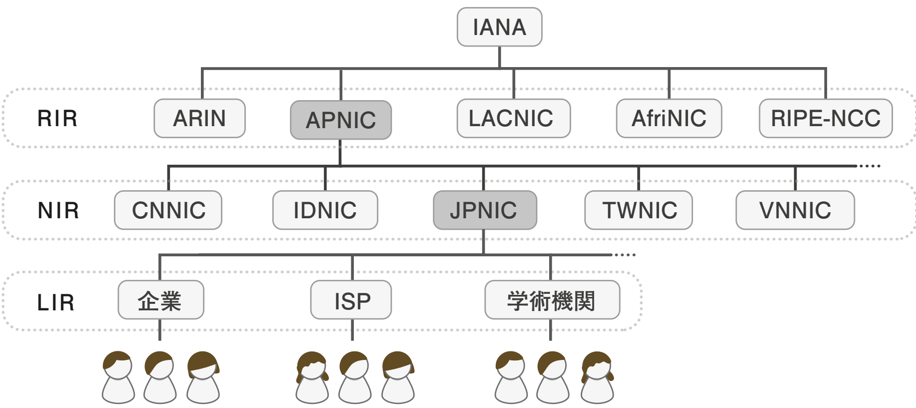
RIRは、LIR（Local Internet Registry、ローカルインターネットレジストリ）などの要求に応じて、IANAから受け取った番号を割り振ります。IANAからIPアドレスの割り振りを受けたLIRは、ユーザからの要求に従って番号の割り当てを行います。

LIRを介さずに、ユーザが直接RIRからIPアドレスの割り当てを受けることもあります。



▶図18.3　LIR

日本を含むアジア太平洋地域には、国など、RIRよりも細かい地域を受け持つNIR（National Internet Registry、国別インターネットレジストリ）が存在しています。日本のNIRは、JPNICです。日本国内のISPなどは、JPNICからIPアドレス割り振りを受ける場合もあれば、APNICからIPアドレス割り振りを受ける場合もあります。



▶図18.4　NIR

このように階層的にIPアドレスを管理することで、複数の異なる組織が同時にまったく同じIPアドレスを「自分が使うためのもの」としてしまうことを避けるような仕組みになっています。

国や地域によって多少制度が異なる部分がありますが、インターネットのアドレスや番号を利用するには、RIRやNIRにお金を払い続ける必要があります。誰でも無制限に番号や資源を使えるわけではなく、申請と登録が必要なのです。

注意が必要なのは、IPアドレスやAS番号を購入する代金ではなく、割り当て業務を支える維持料であるという点です。たとえば、日本国内でJPNICからIPアドレス割り振りを受けるのであれば、JPNICの指定事業者になる必要があります。指定事業者がJPNICに支払う維持料

は、AS番号の契約数やIPアドレスサイズによって変動します。一般ユーザが日々使っているIPアドレスを利用するために、ISPなどはお金を払っています。

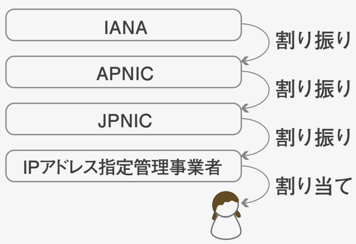
「割り振り」と「割り当て」

本書では、IPアドレスに関して、**割り振り**と**割り当て**という2つの表現を使い分けています。この使い分けは、日本語に特有の使い分けではなく、2つの異なる英語に対応しています。すなわち、Allocationが「割り振り」で、Assignmentが「割り当て」です。

Allocationは、ある特定の範囲を特定の組織に対して「割り振り」することを表します。プログラミングの経験があれば、「メモリのアロケーション」という表現を聞いたことがあるかもしれません。この場合のAllocationには、メモリの特定の領域を確保するという意味があります。IPアドレスのAllocationも、IPアドレスの特定の範囲を、ある特定の組織に対して「割り振り」します。

「割り振り」を受けた組織は、特定のIPアドレスを「割り当て」ます。最終的にはユーザに対してIPアドレスが「割り当て」られるのです。

日本でJPNICからIPアドレス分配を受ける場合、IPアドレス分配の階層構造は、IANA→APNIC→JPNIC→IPアドレス指定管理事業者→ユーザとなっていますが、その中での「割り振り」と「割り当て」は図18.5のようになります。



▶ 図18.5 「割り振り」と「割り当て」

IANA、APNIC、JPNICによる分配が「割り振り」、IPアドレス指定管理事業者による分配が「割り当て」となっており、最終段階だけが「割り当て」です。

IPアドレス指定管理事業者が自分の組織内にあるネットワークでIPアドレスを使う場合にも「割り当て」となります。

18.3.2 段階的に発生するIPv4アドレス在庫枯渇

話をIPv4アドレス在庫枯渇に戻しましょう。IANAとRIRには、それぞれIPv4アドレス在庫があります。世界にいくつかの「IPv4アドレス在庫」が存在している形です。「IPv4アドレス在庫」は、「IPv4アドレスプール」と呼ばれることもあります。それらのIPv4アドレス在庫が枯渇するのが、現実の「IPv4アドレス在庫枯渇」です。

最初に発生するIPv4アドレス在庫枯渇は、中央在庫であるIANAのIPv4アドレス在庫の枯渇です。IANAのIPv4アドレス在庫は2011年2月に枯渇しました。IANAは、32ビットのIPv4アドレス空間を256個のブロックとして管理していました。そのすべてのブロックがIANAのIPv4アドレス在庫からなくなった状態がIANAにおける「IPv4アドレス在庫枯渇」です。

次に発生するのが地域ごとのRIRにおけるIPv4アドレス在庫枯渇です。日本のようにNIRがある地域では、RIRのIPv4アドレス在庫枯渇後に、NIRのIPv4アドレス在庫枯渇が発生します。このように、IPv4アドレス在庫枯渇は段階的、かつバラバラのタイミングで発生します。

IANA在庫の枯渇のアナロジーとしては「IPv4アドレス製造工場で製造中止された状態であって、問屋と小売店には、まだIPv4アドレス在庫が残っている」という感じかもしれません。このとき、RIRやNIRでのIPv4アドレス在庫枯渇は、問屋や小売店のIPv4アドレス在庫が枯渇した状態というアナロジーになります。

18.4 IPv4アドレス在庫枯渇の対策

すでにIPv4アドレスの在庫が枯渇している現状で、IPv4において実施できるのは、実際には解決策ではなく緩和策です。では、IPv6対応とは別の問題としてIPv4アドレス在庫枯渇を捉えた場合、どのような緩和策が考えられるでしょうか。

IPv4アドレスを新たに割り当てられないにもかかわらず、IPv4インターネットの規模を拡大したいとしたら、既存のIPv4アドレスを使い回すしかありません。具体的には、一部のIPv4アドレスの利用を圧縮してほかにまわすといった対応が要求されるようになります。とはいえ、実はそんなに対応可能なことがあるわけではなく、考えられるのは以下のような基本的な対策くらいです。

1. 現在利用しているIPv4アドレス数を圧縮して再利用する
2. 1つのIPv4アドレスで複数サービスを運用して有効利用する
3. 他者からIPv4アドレスを確保する（IPv4アドレス移転など）

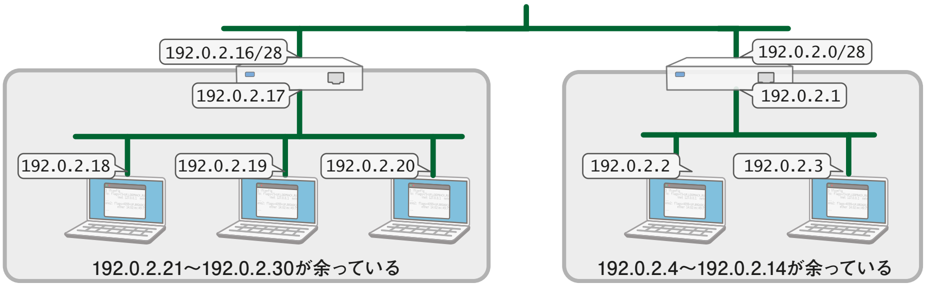
1の圧縮で利用されるのは、CGN（Carrier Grade NAT）などの技術です。IPv4アドレス在庫が枯渇した時点で多くのユーザを抱えていた事業者は、IPv4アドレスも多く保持しています。そのような事業者であれば、ユーザに対して配布しているIPv4アドレスの数をNATを使って圧縮することで、サーバ設置やIaaSの運営にまわすという方法が考えられます。

2の有効利用は、さまざまな理由で活用できていなかったIPv4アドレスをきっちり使い切ることを指しています。運用上の優先度が低かったり、事実上は利用していないにもかかわらずIPv4アドレスが使われてしまっているところから引き上げるというものです。具体的には、終了を予定していた既存サービスなどを早めに切り上げてIPv4アドレスを回収したり、別々のセグメントに分散していた複数の機器を1つのセグメントに集約して余ったIPv4アドレスを回収するという方法があります。枯渇時点で多くのIPv4アドレスを保持していれば、それらを節約してほかにまわすことで、事業を拡大できる余地が多少は広がります。

このように、上記の1と2の対策は、すでにIPv4アドレスを保持していることが前提です。IPv4アドレス在庫が枯渇した時点で保持していたIPv4アドレスのベースが小さければ小さいほど、圧縮や節約の効果も小さくなります。そういうことをできない事業者は、新規IPv4アドレスを捻出するために何らかの方法を模索しなければならず、苦しい思いをするでしょう。

最後の3は、IPv4アドレスの登録者の移転によって必要なIPv4アドレスを調達する可能性です。これには金銭的なやり取り（いわゆる「IPv4アドレス売買」）を伴う場合もあります。

CC BY NC SA



▶ 図 18.6 別々のセグメントに分散している IPv4 アドレス

IPv4 アドレス在庫枯渇問題への短期的解決策

日本時間 2011 年 2 月 3 日深夜に、IPv4 アドレスの中央在庫（IANA 在庫）が枯渇しました。IPv4 アドレス在庫枯渇問題は 1990 年代から議論されていた問題であり、本書の主題である IPv6 はそれに対する長期的解決策として作られました。

一方、やはり 1990 年代には、IPv4 アドレス在庫枯渇問題に対する解決策として以下の 3 つの技術が開発されました。これらは、IPv4 アドレス在庫枯渇問題の発生を遅らせて IPv4 を延命させることを目的とした技術であり、この問題に対する短期的解決策といえます。

- CIDR（Classless Inter-Domain Routing）
- プライベート IPv4 アドレス
- NAT（NAPT）

その後、実際に IPv4 アドレス在庫が枯渇するという状況になるまでには 10 年以上かかっています。これだけの期間を持ちこたえたのは、短期的解決策として作られたこれらの仕組みが非常に効果的だったからといえるかもしれません。

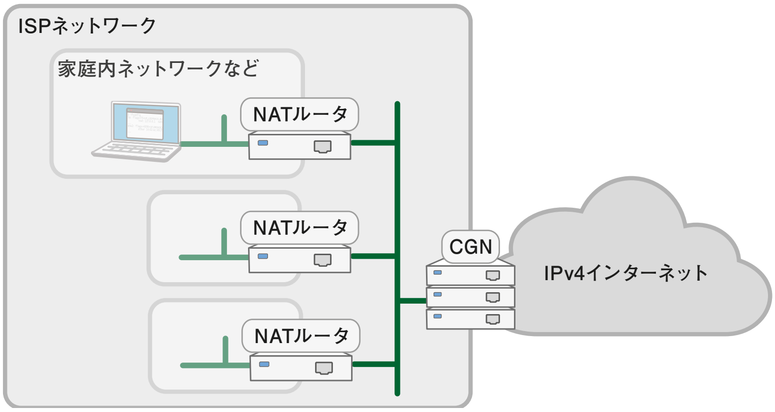
18.4.1 CGN による圧縮

グローバル IPv4 アドレスの利用数を圧縮する代表的な技術として CGN（Carrier Grade NAT）が挙げられます。CGN は、文字どおり、大規模な NAT（Network Address Translation）です。CGN を利用することで、ISP が接続サービスに利用しているグローバル IPv4 アドレス数を圧縮できます。

日本では、1 つの契約に対して 1 つのグローバル IPv4 アドレスを割り当てるサービスを提供する ISP が多くありました。しかし、IPv4 アドレス在庫枯渇に伴い、徐々に CGN の利用が増えています。CGN が導入されると、図 18.7 のように、家庭内の NAT ルータなどにおける NAT に加えてもう 1 段階の NAT が利用されるようになります。

IPv4 NAT と CGN に関しては、第 19 章で詳しく解説します。

CC BY NC SA



▶ 図18.7　一般家庭へのCGN

NOTE

CGNは、利用されるIPv4アドレスの数を圧縮する技術であり、IPv6対応に直接寄与する技術ではありません。利用されるIPv4アドレスの数を圧縮するとともにIPv6対応を徐々に進めるためにCGNを用いる手法については、24.5節で説明します。

18.4.2　単一のIPv4アドレスで仮想的に複数のサービスを提供する

IPv4アドレス在庫が枯渇して時間が経ってくると、ユーザ側だけではなく、サーバ側に対しても徐々に変化が求められるようになるでしょう。特に、個々のサービスを提供するサーバに対して個別にIPアドレスを割り振ることが難しくなっていくと予想されます。そこで考えられるのが、1つのIPアドレスで複数のサーバを仮想的に運用するような技術です。

単一のIPアドレスを利用して仮想的に複数のサービスを利用する方法としては、TCPやUDPのポート番号によるサービスの切り替えが考えられます。たとえば、サーバのIPアドレスが`192.0.2.2`である場合、HTTPのウェルノウンポートである80番で運用されたWebサーバであれば`http://192.0.2.2/`のようなURLでアクセスが可能です。もし、同じIPアドレスで別のWebサーバをTCPの10080番ポートで運用していたら、`http://192.0.2.2:10080/`として別のWebサーバにアクセスできます。このように複数のTCPポート番号を利用することで、単一のIPアドレスを利用しつつ複数のWebサーバプロセスを実行できます。

ただし、クライアント側で明示的にポート番号を指定して通信を開始することは、特にWebのような一般のサービスでは必ずしも現実的ではありません。環境によっては、ファイアウォールなどで宛先ポート番号が制限されていて、一般ユーザが直接外部のネットワークと通信できるのは80番（HTTP）と443番（HTTPS）のみという場合もあります。そういったファイアウォール設定をしている組織内のユーザにとって、80番と443番以外のポート番号で運用されているWebサーバは、存在しないのと同じです。そのため、本書執筆時点においては、商用サービスとしてサーバでウェルノウンポート以外で積極的にサービスを提供している事業者は非常に少ないと思われます。

実際に広く利用されているのは、名前に基づくバーチャルホストを利用することで単一のIPv4アドレスで仮想的に複数のWebサービスを運用する方法です。この方法では、複数の

バーチャルホストが、単一のIPアドレスとTCPポート番号を共有します。たとえば、HTTPのウェルノウンポートである80番を、複数のバーチャルホストで共有できます。複数の契約者が1つのサーバを共用するホスティングサービスなどでも、この方法がよく使われています。

名前に基づくバーチャルホストで注意が必要なのは、HTTPSを利用するときです。HTTPSで利用されるTLSプロトコルでは、接続を希望するホスト名をクライアント側からサーバに伝えるために、SNI（Server Name Indication）というTLS拡張が必要です。TLSサーバは、SNIに未対応のクライアントに対しては、あらかじめ決められたホスト名に紐づけられた証明書を利用して通信を行います。そのため、SNIに未対応のクライアントが多く存在する環境でHTTPSによるサービスを提供するときには、証明書ごとに個別のIPv4アドレスが必要になってしまいます。

18.5 IPv4アドレス移転、IPv4アドレス売買、IPv4アドレス市場

IPv4アドレスの中央在庫が枯渇し、RIRなどの在庫も枯渇すると、インターネットに接続しようとする組織が新たなIPv4アドレスの割り当てを受けられなくなってしまいます。しかし、IPv4アドレスを必要とするサービスは引き続き数多く存在しているので、各組織は何とかしてIPv4アドレスを確保しようとします。

組織がすでに保持しているIPv4アドレスの圧縮や有効利用では間に合わない場合の方策として考えられるのは、他の組織からのIPv4アドレスの移転です。ここでは、他の組織に割り当てられたIPv4アドレスを移転する制度であるIPv4アドレス移転や、いわゆる「IPv4アドレス売買」「IPv4アドレス市場」について紹介します。

18.5.1 IPv4アドレス移転の仕組みが成立する背景

IPv4アドレスを組織間で移転することは、IPv4アドレス在庫の枯渇が間近に迫るまでは、公式には認められていませんでした。とはいえ、IPv4アドレスが貴重な資源になって価値が発生し、組織間でIPv4アドレスの「売買」が行われるだろうことは、誰の目から見ても明らかでした。IPv4アドレスを組織間で移転できる仕組みがないままでは、どうしてもIPv4アドレスが必要な組織は、事業を諦めるかIPv4アドレスを闇取引するしかありません。

IPv4アドレスが闇で取り引きされてしまうと、あるIPv4アドレスを誰が管理しているのかわからなくなるという問題が生じます。現在のインターネットでは、「誰がどのIPアドレスブロックを利用しているか」が中央で管理されており、何か問題が発生したときに管理者がIPv4アドレスをもとに原因となった組織を調べて連絡できる体制が整えられていますが、それが機能しなくなってしまいます。IPv4アドレスを統括的に管理している組織を通さずに、IPv4アドレスの闇取引が横行すると、各IPv4アドレスを実際に利用している組織が把握できない状態が多発して、微妙なバランスで成り立っているインターネットが根底から変化してしまうおそれがあるのです。

そのため、それまでのようにIPv4アドレスの移転を禁止するのではなく、手続きを経てIPv4アドレスを移転できるようにするための仕組みが用意されました。日本が参加しているAPNIC

CC BY NC SA

でも、2010年にIPv4アドレス移転のための仕組みが実装され、組織間でIPv4アドレスを移転できるようになっています。

ここで重要なのは、いわゆる「IPv4アドレス市場」や「IPv4アドレス売買」と、ここで整備されたIPv4アドレス移転の仕組みとは、同じものでないという点です。IPv4アドレス移転では、IPv4アドレスを新規に受け取る団体に対する審査が従来の新規割り当てと同様に存在します。IPv4アドレスを自由に取り引きできるような市場とは違うので、投資目的でIPv4アドレスを取得するような行為は困難であると考えられています。

また、仮にIPv4アドレスの取引市場が確立したとしても、IPv4アドレスそのものの供給は簡単に増やせません。IPアドレスに対する需要は、IPv4アドレス枯渇後も、世界的に増加し続けます。その一方で、使い続けることが前提のIPv4アドレスがそのような市場に供給される量は減少していくばかりでしょう。最初のうちは、利用していないIPv4アドレスを市場で売り出す組織が登場する可能性もありますが、そこで購入されたIPv4アドレスが再度流通する可能性はほとんどないと考えられます。時間が経過すればするほど供給が減少し、IPv4アドレスの価格は高騰していく可能性があります。価格が高騰することで売り手として前向きになる組織が増える可能性もありますが、IPv4アドレスを売るということは、自分のネットワーク規模を縮小するということです。どの組織も一定以上は身を削れないので、価格が上昇したからといって供給の継続的な増加は見込めないと考えられます。

18.5.2 Final /8 Policy

IPv4アドレス在庫が枯渇するまでは、IPv4アドレス移転や、いわゆる「IPv4アドレス売買」と呼ばれる金銭的な見返りを伴うIPv4アドレス移転に対して否定的な見方が大勢でした。しかし、JPNICおよびAPNICでのIPv4アドレス在庫枯渇と、IANAの中央在庫から各RIRへの最後の5つの/8ブロックの割り当てによって、雰囲気が大きく変わりました。

各RIRへ割り当てられた最後の5つの/8ブロックは、その分配方法について、それまでのブロック（通常在庫）とは違う特別なポリシーが定められています。これはFinal /8 Policyと呼ばれています。RIRで通常在庫が枯渇すれば、この最後の5つのブロックから分配されることになり、Final /8 Policyが発動します。これにより、各組織は最後に一度しかIPv4アドレスの割り振りを受けられなくなり、IPv4アドレスが必要になってもこれまでのようには申請できなくなりました。新たにIPv4アドレスが必要になった場合、IPv4アドレス移転以外の方法が完全になくなったのです。

さらに、Final /8 PolicyによってIPv4アドレス返却の意味が変わったという側面もあります。それまでは、返却されたIPv4アドレスは、しかるべき期間が過ぎたあとに再割り振りされていました。しかし、Final /8 Policy発動後は、割り振りを受けられる組織が限定されると同時に、一度に受けられる割り振りサイズも小さくなってしまったので、返却されるIPv4アドレスが死蔵される状況となりました。死蔵されるならば移転したほうが有効利用であるという意見が増えたことで、IPv4アドレス移転に対する考え方も変わりました。

cc BY NC SA

IPv4アドレスの返却は焼け石に水

2010年頃は、IPv4アドレス在庫枯渇の話題に関連して、IPv4アドレスを保持している組織に対して「たくさん持ってるんだから返せ」という意見がネット上などで多数登場しました。しかし、たとえ複数の企業が持っているIPv4アドレスを返却したとしても、それは焼け石に水でしかなく、実はあまり建設的ではありません。

IANAのIPv4アドレス中央在庫が枯渇する前の2010年時点でのIPv4アドレス割り振りペースは、IPv4アドレス空間全体の1/256の大きさである/8ブロックが約1ヶ月で割り当て終わってしまうという速度でした。2010年の段階で、それだけインターネットが急激に成長していたことを示しています。そこからざっくり逆算すると、/8ブロックの1/256である/16ブロックには、成長を続ける世界のインターネットの3時間分の需要しか満たせていなかったことになります。つまり、ある組織が何ヶ月、あるいは何年もかけて組織内ネットワークの構成変更を準備して/16ブロックを返却したとしても、一瞬でそのすべてが割り当て終わってしまいます。IPv4アドレスを少し余分に持っている組織から返却があったとしても、砂漠にバケツ一杯の水を撒くような状態になってしまうという状況だったのです。

18.5.3 IPアドレスの維持料

IPv4アドレス移転を理解するうえでは、IPアドレスには維持料がかかるという事実も重要です。JPNICにおけるIPアドレス維持料は、IPv4のPAアドレス（Provider Aggregatable Address）の場合、/24で年間55,000円、/16で年間448,652円です。

さらに、2012年4月からは、それまで課金されていなかった歴史的PIアドレス（Provider Independent Addresses）に対しても課金が実施されるようになりました。それまで課金されていなかった歴史的PIアドレスを利用していた日本国内の大学などでは、経費削減のためにIPv4アドレス移転を検討するようになりました。

18.5.4 IPv4アドレス移転と経路爆発問題

IPv4アドレス移転において、IPv4アドレスを譲渡する側の組織は、自分が保持しているIPv4アドレスブロックから自分が使っていない部分を切り出して渡します。すると、それまで1つだった経路が複数に分割されてしまいます。

IPv4アドレスの売買が盛んになるようなことがあれば、大きなアドレスブロックを持つ組織がIPv4アドレスを細切れにして多数の組織に売り渡し、それだけ大量の経路がインターネット上にあふれる可能性があります。細切れになって販売されたIPv4アドレスを購入した結果、同じ組織が連続していない細切れのIPv4アドレスを利用する状態が発生する可能性もあります。IPv4アドレスの売買が世界各地で活発になると、それまでにない勢いでインターネット上で経路数が増加し、処理性能が低い（メモリが少ない）古いルータが処理しきれなくなって一部の経路への到達性を失うネットワークが発生するかもしれません。このため、IPv4アドレスの過度に自由な売買が行われるようになれば、結果としてインターネットを不安定にしてしまうかもしれないと考えられています。

cc BY NC SA

NOTE

細切れのブロックによる経路爆発問題は、IPv4アドレス移転に限った話ではありません。2011年以降、RIRやNIRなどでIPv4アドレス在庫枯渇が発生し、最小割り振り単位が小さくなったことによって、それまで以上に細かい経路が増えているという側面もあります。

18.5.5　IPv4アドレス移転に関する情報

APNIC管理下にあるIPv4アドレスについては、2011年1月から移転が可能になりました。さらに、2011年8月1日には、JPNIC管理下にあるIPv4アドレスの移転も可能になりました。2011年の段階ではIPv4アドレスの移転はRIR内のみに限定されていましたが、2012年にはRIRを越えた移転もできるようになりました。

IPv4アドレス移転に関する情報は、以下のURLで公開されています。

- JPNIC：`https://www.nic.ad.jp/ja/ip/ipv4transfer-log.html`
- APNIC：`http://ftp.apnic.net/transfers/apnic/`

ただし、これらの公開情報はAPNICやJPNICが直接かかわっている移転そのものに関するものであり、移転に伴う当事者間での金銭授受や契約などに関する情報は範疇外となっています。

18.5.6　技術的課題以外の部分が多いIPv4アドレス在庫枯渇対策

IPv4アドレス在庫枯渇に対する直接的な対策は、初期段階においてはIPv4アドレス移転（およびそれに伴う金銭的やり取り）が主な手法となるでしょう。しかし、移転可能なIPv4アドレス保持者が徐々に減っていくことを考えると、IPv4アドレス移転は継続的な効果がある手法ではありません。最終的には、IPv4アドレスを保有する企業が倒産するなどのタイミングでしか、まとまったIPv4アドレスが放出されなくなるでしょう。

当面、最も活用される手法は、IPv4アドレスの利用数を圧縮するためのCGNになると思われます。ただ、CGNにはさまざまな課題もあり、ユーザにとって不便な環境が増える可能性もあります。

IPv4アドレス在庫枯渇に対する直接的な解決手法は非常に限られています。一方で、インターネットは成長を続けており、IPv4によるインターネットは徐々に複雑化していきます。複雑化するだけではなく、IPv4アドレスをすでに保有する組織と保有していない組織の格差も時間の経過とともに拡大していくことが予想されます。このままIPv4アドレスが足りない状態で進んでいくと、IPv4上でのサーバ運用が複雑化すると同時に、それらを制御するためのプログラムの開発、テスト、デバッグにかかる工数も上昇するでしょう。複雑なネットワークには管理コストもかかります。

ネットワーク運用者やインターネットでサービスを提供する事業者にとって、IPv4アドレス在庫枯渇に伴う影響がないIPv6へとユーザを移行させるメリットはここにあります。繰り返しになりますが、IPv6はIPv4アドレス在庫枯渇に対する解決策にはなりません。しかし、IPv6ネットワークの環境を整備し、IPv6対応を進めていかなければ、インターネットそのものがいまよりもさらにいびつな形になっていくでしょう。

BY NC SA

第19章

IPv4アドレス共有技術

前章では、IPv4 アドレス在庫枯渇問題に対する短期的解決策としてIPv4 NAT（Network Address Translation）の存在を挙げました。いまやIPv4 NATはIPv4アドレス資源を有効活用する手法として世界中で広く使われています。IPv4アドレス在庫枯渇が発生する数年前からは、ISPなどで利用するIPv4アドレス利用数を圧縮するために、CGN（Carrier Grade NAT）など、大規模なIPv4 NATによるIPv4アドレス共有技術の利用も広がりました。IPv4アドレス共有技術は、IPv4ネットワークでのCGNだけではなく、DS-LiteやMAP-EなどのIPv4 over IPv6トンネルに関連する技術や、NAT64によるIPv4とIPv6の変換技術を運用する際に利用されることもあります。

IPv4アドレス共有技術の利用が増えることで、インターネットにおけるIPv4での通信品質が劣化すると表現されることもあります。この「IPv4の劣化」が、IPv4アドレス在庫枯渇問題への長期的な対策としてIPv6普及が求められる背景のひとつであるとする考え方もあります。

現在のインターネットは、IPv4アドレス在庫枯渇問題に対し、IPv4 NATによる短期的な解決策とIPv6による長期的な解決策という2種類の解決策が併用されている状況です。本章では、IPv4インターネットとIPv6インターネットが両立している現在のインターネットの姿を伝えるために、IPv4アドレス在庫枯渇問題のもう1つの解決策であるIPv4 NATについてある程度詳しく説明します。

19.1 NATとプライベートIPv4アドレス

NATは、途中経路上の機器で、パケットが持つIPアドレスを別のIPアドレスに変換する仕組みです。NATの主な用途は、ISPから提供される1つのグローバルIPv4アドレスを利用して、プライベートなネットワーク内で複数の機器をインターネットに接続するというものでした。ISPがユーザに対してIPv4を割り当てるときには、一般にサブネット単位で割り当てるのではなく、単一のIPv4アドレスを割り当てます。そういった環境で、複数の機器をインターネットに接続したい場合、個々の機器のIPv4アドレスをISPから割り当てられた単一のIPv4アドレスに変換するためにNATが必要になります。

(CC BY-NC-SA)

NATに関する最初のRFCはRFC 1631[†1]です。このRFCで説明されているもともとのNATは、IPアドレスだけを変換するものでした。しかし現在のNATでは、TCPとUDPのポート番号を考慮して、単一のIPアドレスを複数のユーザが同時に利用できるようになっています。これを特に**NAPT**（Network Address Port Translation）と呼んで区別する場合もよくあります。実際、2001年に発行されたRFC 3022[†2]では、RFC 1631で定義されていたNATを「Basic NAT」と呼び、Basic NATとNAPTを合わせて「Traditional NAT」と表現しています。ただし、昨今ではNAPTのことを含んでNATと表現することが多いので、本書でもNATといったら基本的にNAPTを含むものとします。

NOTE

NATに関連する用語定義については、1999年に発行されたRFC 2663[†3]も参照してください。

19.1.1　NATがIPv4インターネットで広く普及した背景

NATそのものは、すでに何度か触れているように、IPv4アドレス在庫枯渇問題を強く意識して作られた仕組みです。そのためインターネットが作られた当初は存在していませんでした。NATが開発されて広く利用されるようになったのは、インターネットが一般家庭などにまで急速に普及し始めた1990年代以降です。

ただし、NATが普及した要因はIPv4アドレス在庫枯渇問題そのものではなく、むしろユーザのニーズにマッチしたことでした。具体的には、「ISPから提供される1つのグローバルIPv4アドレスを利用して、プライベートなネットワーク内で複数の機器をインターネットに接続する」という用途です。

NATがIPv4アドレス在庫枯渇問題に対する短期的な解決策として効果的だったのは、ユーザのニーズにマッチした技術として普及したからこそだといえます。そこで、このニーズについて理解するために、1990年代前半の日本における商用インターネット接続サービスの状況を簡単に振り返ってみましょう[†4]。

1990年代前半の日本では、一般家庭からのインターネット接続において、主に電話線を利用したダイアルアップ接続が利用されていました。1つの電話線でインターネットに接続できる機器は1つだけでしたが、当時は複数の機器をインターネットに接続する必要がある家庭はほとんどありませんでした。しかし、ISDNやADSLといったインターネット接続サービスの登場やパソコンの普及に伴い、家庭内の複数の機器から同時にインターネットに接続したいというニーズも次第に増えていきます。このニーズを満たすためには、同時にインターネットに接続したい機器のそれぞれにグローバルIPv4アドレスが必要でした。

---

†1　RFC 1631：K. Egevang, P. Francis, “The IP Network Address Translator (NAT)”, 1994年5月

†2　RFC 3022：P. Srisuresh, K. Egevang, “Traditional IP Network Address Translator (Traditional NAT)”, 2001年1月

†3　RFC 2663：P. Srisuresh, M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations”, 1999年8月

†4　ここでは日本における状況について説明しますが、「NAT普及の背景にあったユーザのニーズ」という視点では海外の状況もそこまで大きな違いはありません。

ユーザは、自分が契約しているISPからIPv4アドレスを割り当ててもらいますが、このとき一般的なISPのサービスがユーザに割り当てるのは「単一のIPv4アドレス」です。そのためユーザは同時に1つの機器しかインターネットに直接つなげられません。複数の機器を同時にインターネットで利用するには、サブネット単位でIPv4アドレスを割り当ててもらうなどの特別な契約や別の回線が必要でした。

このような状況で「ISPから割り当てられた単一のグローバルIPv4アドレスを使って複数の機器を同時にインターネットに接続する」ための技術として登場したのがNATです。ISPから割り当てられた単一のグローバルIPv4アドレスをNATルータに設定し、家庭内にある個々の機器はプライベートIPv4アドレスでインターネットに接続します。インターネットから見ると直接接続している機器はNATルータだけですが、そのNATルータがプライベートIPv4アドレスを変換することで複数の機器をインターネットに接続できるようになります。1つのIPv4アドレスという限られた資源を、NATルータを介して複数の機器で共有するという形態です。

さまざまな呼び名がある家庭用ルータ

本章ではIPv4 NATの機能を実現する簡易な家庭用ルータを**NATルータ**と表現しています。このような家庭用の簡易なNATルータは、RFCなどでは**CPE**（Customer Premises Equipment）と呼ばれていることがあります。なお、製品としては「SOHOルータ」や「ブロードバンドルータ」、あるいは無線LANのアクセスポイントとしても機能する「無線LANルータ」という名前で販売されている場合もあります。

19.1.2 プライベートIPv4アドレス

前項では、NATルータの普及の背景として、「ISPから割り当てられた単一のグローバルIPv4アドレスで複数の機器を同時にインターネットに接続する」というニーズがあったことを説明しました。この構成ではNATルータの内側で**プライベートIPv4アドレス**が利用されます。プライベートIPv4アドレスは、家庭内のほか、マンション全体の共用回線、オフィスにおけるLAN、スマホなどでのテザリングなど、いまでは当たり前のように閉じた環境で幅広く利用されています。

プライベートIPv4アドレスは、RFC 1597[†5]において、以下の3つのIPv4アドレスのブロックとして定義されています。

- 10.0.0.0 〜 10.255.255.255（10.0.0.0/8）
- 172.16.0.0 〜 172.31.255.255（172.16.0.0/12）
- 192.168.0.0 〜 192.168.255.255（192.168.0.0/16）

プライベートIPv4アドレスが送信元や宛先になっているIPv4パケットはインターネットにおける転送が禁止されています。そのため、インターネットにプライベートIPv4アドレスを送

†5 RFC 1597：Y. Rekhter,B. Moskowitz,D. Karrenberg,G. de Groot, “Address Allocation for Private Internets”, 1994年3月

CC BY NC SA

信元とするパケットが転送されても、そのパケットに対する応答が送信者まで届くことはありません。プライベートIPv4アドレスは、あくまでも閉じたネットワークで利用されることを前提としたアドレスであり、インターネットで扱ってはいけないことになっているからです。

プライベートIPv4アドレスもまたNATと同様にインターネットが誕生した当初は存在していませんでした。プライベートIPv4アドレスがIANAに予約されたことを示すRFC 1597は1994年3月に発行されたものであり、途中で追加された仕様であることがわかります。なお、NATに関する最初の仕様であるRFC 1631が発行されたのは1994年5月で、わずか2ヶ月後です。実際、NATとプライベートIPv4アドレスは、IETFにおいて同時に議論されていました。両者の2ヶ月の差は、RFCとして発行されるまでの議論や準備に伴う差にすぎず、両者は事実上同時に成立した仕組みだといえます。

プライベートIPv4アドレス以前の閉じたTCP/IPネットワーク

プライベートIPv4アドレスが登場する以前も、インターネットと直接接続しない閉じた環境でTCP/IPを使う通信システムは存在していました。通信手段としてTCP/IPを利用する以上、そうしたシステムでも、それぞれの通信機器に対して何らかのIPアドレスを割り当てる必要があります。当時は、そのような閉じた環境であっても、世界で一意のIPv4アドレスが利用されていました。

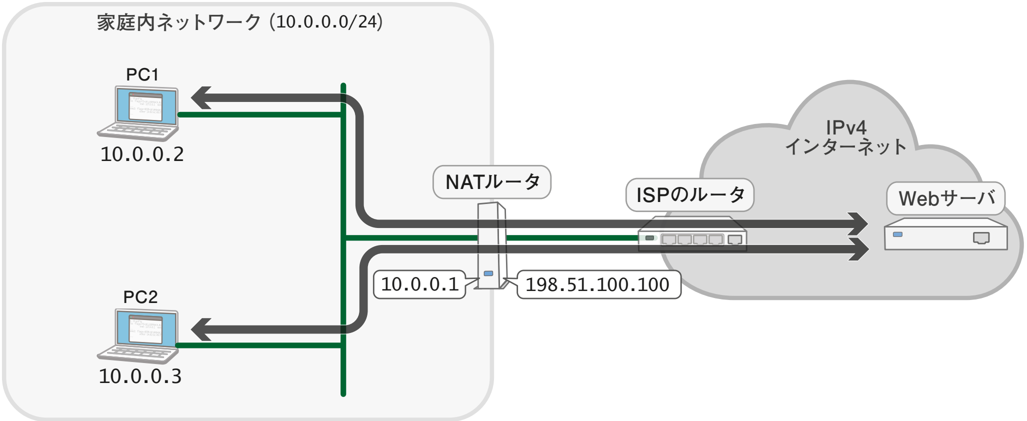
しかし、インターネットと通信するわけではない閉じた環境における通信であれば、その組織内においてのみ一意性が保たれるIPv4アドレスを使えば十分です。インターネットと接続しない閉じた環境のために、世界で一意のIPv4アドレスが消費されていけば、いつかIPv4アドレスが足りなくなってしまいます。そこで、「閉じた環境であれば自由に使ってよいIPv4アドレスブロック」として生まれたのがプライベートIPv4アドレスです。

IANAにおいてIPv4アドレスの中央在庫が実際に枯渇したのは2011年のことですが、もしプライベートIPv4アドレスが存在しなかったならもっと早く枯渇が起きていたことでしょう。プライベートIPv4アドレスを利用したNATは非常に効果的なIPv4アドレス在庫の延命策であったと考えられます。

19.2　一般的なIPv4 NATの仕組み

NATの典型的なユースケースを図19.1に示します。図19.1の例では、NATルータのインターネット側のインターフェースに対し、グローバルIPv4アドレス`198.51.100.100`が割り当てられています。一方、家庭内ネットワークではプライベートIPv4アドレスとして`10.0.0.0/24`を利用しています。そして、NATルータの家庭内ネットワーク側のインターフェースには`10.0.0.1`、PC1には`10.0.0.2`、PC2には`10.0.0.3`というプライベートIPv4アドレスをそれぞれ割り当てています。プライベートIPv4アドレスの家庭内ネットワークに接続された複数の機器が、NATを利用して1つのグローバルIPv4アドレスを共有していることがわかります。

cc BY NC SA



▶ 図19.1 NATの典型的なユースケース

19.2.1 NATルータでの処理

NATルータでは、それぞれのパケット情報の変更をどのように行うべきかを示すマッピングテーブルを作成し、同じフローに属するパケットが同じフロー[†6]として扱われるように変換します。グローバルIPv4アドレスで運用されたインターネット側からのパケットがNATルータへと到達すると、マッピングテーブルに応じて変換が行われ、プライベートネットワーク内の適切な機器へとパケットが転送されるように処理されます。

この処理を、図19.1のような環境でNATの内側にいるPC1からインターネット上のWebサーバに接続する状況を例に、少し詳しく見てみましょう。

まず、NATルータの内側のネットワークについて状況を整理します。このネットワークは10.0.0.0/24というプライベートIPv4アドレス空間で、それぞれの機器には次のようなプライベートIPv4アドレスが設定されています。

- NATルータの内側のインターフェースのIPv4アドレス：10.0.0.1（PC1のデフォルトゲートウェイになる）
- PC1のIPv4アドレス：10.0.0.2

一方、NATルータのインターネット側のインターフェースには、ISPから割り当てられたグローバルIPv4アドレスが設定されています。

- NATルータのインターネット側のインターフェースのIPv4アドレス：198.51.100.100

このような状況で、PC1からグローバルIPv4アドレス203.0.113.30のWebサーバへと接続するとします。このときPC1から最初に送信されるのは、WebサーバのTCP 80番ポートに接続するためのTCP SYNパケットです。例として、このTCP SYNパケットの送信元ポート番号は1111番だったとしましょう。まとめると、PC1からWebサーバへのTCP SYNパケットには次のような情報が設定されているとします。

†6 送信元と宛先IPアドレス、送信元と宛先ポート番号、トランスポートプロトコルの種類という5つの情報の組で区別される1つの通信のことです。

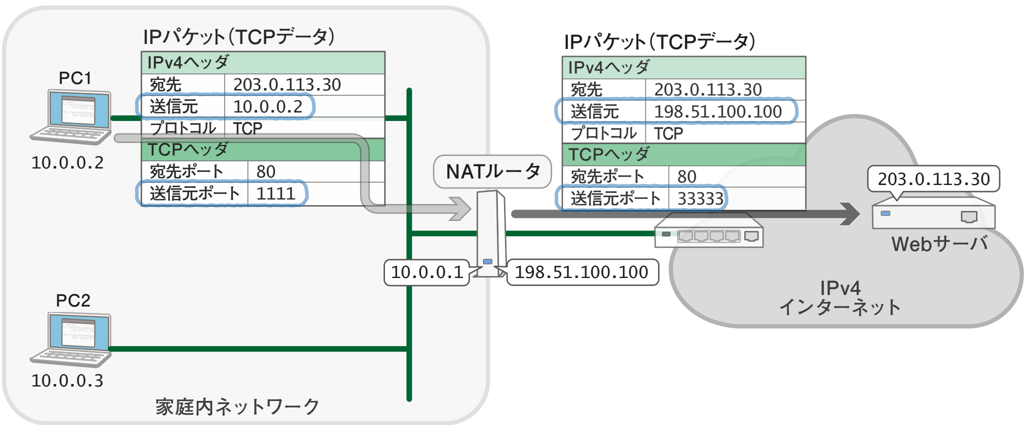
cc BY NC SA

- 宛先IPv4アドレス：203.0.113.30
- 送信元IPv4アドレス：10.0.0.2
- 宛先ポート番号：80
- 送信元ポート番号：1111

PC1からWebサーバAに対するTCP SYNパケットは、デフォルトゲートウェイ（NATルータの内側のインターフェース）へと送信されます。NATルータはこのTCP SYNパケットを受け取ると、TCP SYNパケットの送信元IPv4アドレスを10.0.0.2から198.51.100.100へと変更します。また、TCP SYNパケットの送信元TCPポート番号も変更します。ここでは仮に1111から3333へと変更したとします。

- 送信元IPv4アドレス：10.0.0.2→198.51.100.100
- 送信元ポート番号：1111→3333

ここまでの状況を図19.2に示します。



▶図19.2　PC1からWebサーバへの通信

このNATルータによる変換の結果、「NATの内側での送信元IPv4アドレス」と「NATルータを経由したインターネット側での送信元IPv4アドレス」の組み合わせなどの対応関係を表す情報が決まります。この組み合わせ情報を作ることはNAT bindingと呼ばれます[†7]。そしてNATルータでは、それらの組み合わせ情報を、NATテーブルと呼ばれる情報のエントリとして保持します。

いまの例では、NATテーブルに次のようなエントリが記録されることになります。

[†7] NAT bindingは、RFC 3424、RFC 4787、RFC 5780などで使われている用語で、「結合する」、「組み合わせる」、「束ねて縛る」などの意味を持った“bind”という英単語に由来する表現です。「どのように対応づけられているか」という文脈ではNAT mappingと表現されることもあります。

NATの内側での宛先IPv4アドレスとポート	203.0.113.30:80
NATの内側での送信元IPv4アドレスとポート	10.0.0.2:1111
NATの外側での宛先IPv4アドレスとポート	203.0.113.30:80
NATの外側での送信元IPv4アドレスとポート	198.51.100.100:3333

NATルータで変換されたTCP SYNパケットは、NATルータのインターネット側のインターフェースに設定された198.51.100.100というIPv4アドレスを送信元とするパケットとして、Webサーバに到着します。これを受け取ったWebサーバでは、TCP SYNパケットの接続を許可するために、TCP SYN+ACKパケットを返信します。このTCP SYN+ACKパケットの送信元IPv4アドレスは203.0.113.30、宛先TCPポート番号は3333とします。

Webサーバからこの TCP SYN+ACKパケットを受け取ったNATルータは、NATテーブルのエントリに基づいてPC1宛のパケットに変換し、NATの内側へと転送します。こうしてPC1は特に何も気にすることなくWebサーバと通信できます。

NATテーブルに保持される情報

多くの場合、NATテーブルには以下のような情報が保持されます（NATルータの実装に依存します）。

- TCP、UDP、ICMPといったプロトコルの種別
- NATの内側から見た送信元IPアドレスおよびポート番号（内部ローカル）
- NATの内側から見た宛先IPアドレスおよびポート番号（外部ローカル）
- NATの外側から見た送信元IPアドレスおよびポート番号（外部グローバル）
- NATの外側からの宛先IPアドレスおよびポート番号（内部グローバル）

UDPとTCPのセッションを識別する5つの情報である、送信元IPアドレス、宛先IPアドレス、送信元ポート番号、宛先ポート番号、トランスポート層プロトコルを「5タプル」（5 tuple）と表現することもありまが、NATテーブルにおいても5タプルが利用されます。

19.2.2 内側から別の通信が開始された場合

PC1とWebサーバがNATルータ経由で通信している状況で、さらにPC2もWebサーバとの通信を開始した場合についても考えてみましょう。PC2からWebサーバへの送信元TCPポート番号は2000だとします。

PC2からWebサーバに向けたTCP SYNパケットを受け取ったNATルータは、その送信元IPv4アドレスを10.0.0.3から198.51.100.100へと変更します。また、TCP SYNパケットの送信元TCPポート番号も変更します。ここでは2000から2001へと変更するとします。PC1からの通信のときとは異なる送信元ポート番号に変更していることに注目してください。

- 送信元IPv4アドレス：10.0.0.3→198.51.100.100
- 送信元ポート番号：2000→2001

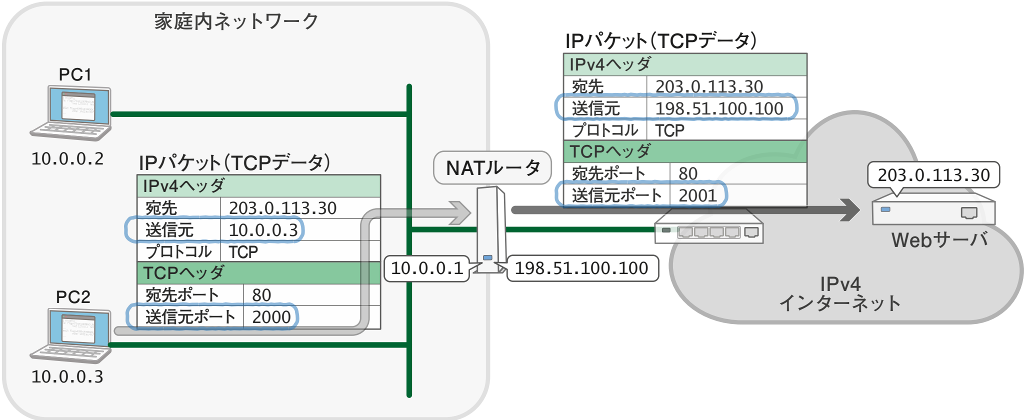
cc BY NC SA

この変更についてもNATテーブルに記録が作成されます。

NATの内側での宛先IPv4アドレスとポート	10.0.0.3:80
NATの内側での送信元IPv4アドレスとポート	10.0.0.1:2000
NATの外側での宛先IPv4アドレスとポート	203.0.113.30:80
NATの外側での送信元IPv4アドレスとポート	198.51.100.100:2001

変更後の送信元ポート番号がPC1のときとは異なるので、このTCP SYNパケットに対してWebサーバから返信されるTCP SYN+ACKパケットにおいても、宛先TCPポート番号がPC1のときとは異なるのがポイントです。具体的には、今度は宛先ポート番号が2001になっているので、NATルータではNATテーブルを参照してPC2宛として変換すればいいとわかります。

ここまでの状況を図19.3に示します。



▶図19.3　PC2からWebサーバへの通信

このようにNATルータでは、内部から外部への通信に際してアドレスとポート番号の変換を同時に行ってその対応をNATテーブルに保持しておき、その情報を使うことで1つのグローバルIPv4アドレス宛に外部から送られるパケットを内部の適切な機器へと転送処理します。これにより1つのIPv4アドレスを複数台で有効に利用する仕組みがNATというわけです。

19.2.3　NATの外側では、内側からの通信がどう見えるか

最後に視点を変えて、NATの内側から外側への通信を、外側のノードからの視点で見てみましょう。いまの例で、Webサーバからは、同じNATルータを利用する2台のPCからの通信がどのように見えるでしょうか。

言われてみれば当たり前だと思いますが、Webサーバからは「203.0.113.30という1つのグローバルIPv4アドレス」から2つのTCPのセッションが張られているようにしか見えません。つまり、PC1とPC2という別々の機器からの通信には見えません。NATルータの存在により、実際に自分に接続している端末が判別しにくくなっているといえます。これはまた、インターネットを流れているIPv4パケットが「NATルータを経由したものかどうか」が判別できないともいえます。

CC BY NC SA

チェックサムの再計算

「NATルータで送信元IPv4アドレスと送信元ポート番号が変更される」というのは、必ずしも「IPパケットのヘッダでこれらのフィールドだけが変更される」という意味ではありません。IPヘッダとTCPヘッダには、パケットに含まれるデータが途中で破損していないかを確認する目的で、パケットの内容に対して計算したチェックサムを格納するフィールドがあります。NATルータはパケットを書き換えるので、このチェックサムフィールドも再計算した値に更新しなければなりません。

なお、IPv4ヘッダに含まれるチェックサムは、通常のルータがIPv4パケットを転送してTTLの値が1減算される場合を含め、IPv4ヘッダに変更が発生するたびに再計算されます。NATルータにおいてIPv4ヘッダを変更するときも、当然チェックサムの再計算が必要になるわけです。

19.2.4 NATテーブルからのエントリの削除

NATルータは物理的な機器であり、記録などに使える資源は有限です。そのため、内側からの通信で生成したNATテーブルのエントリをいつまでも保持するわけではありません。通信がまったく行われていないセッションに割かれている資源は、いつか解放する必要があります。

一見すると、通信セッションが完了したタイミングでエントリを削除すればよさそうに思えるかもしれません。しかし、インターネットでは通信セッションが突然途切れることもあるので、「確実にセッションが切れた」ことを判断できない状況もかなりあります。そのためTCPでは、いったん接続が成立すると、相手が突然音信不通になっても一定の期間はTCP接続状態を維持することになっています。たとえば、Webサイトを閲覧中にパソコンの通信を切断して電源も落とすと、Webサーバに「TCP接続を終了する」という連絡がくることはないので、Webサーバでは通信相手が復帰した場合に備えてTCP接続を維持し、一定の時間が経過して初めてTCP接続を破棄します。

このような状況で、途中経路にあるNATルータでは、WebサーバにおいてTCP接続の状態が破棄された瞬間を知ることができません。NATルータの内側と外側で通信の両端が明示的にTCP接続を終了せずにTCPセッションが消えてしまったような場合、NATルータのNATテーブルには使われることがないゴミが残ってしまうのです。

さらに、インターネットではTCPだけでなくUDPも使われています。UDPには、そもそもTCPのSYNやFINやRSTのような通信セッションの開始や終了を明確に示す仕組みがありません。

これらの理由のため、NATルータでは、NATテーブル中の利用されていない通信セッションのエントリを削除するタイミングを自身で決める必要があります。「この通信セッションは破棄された」と判断するタイミングを「生存時間」や「lifetime」と呼びます。個々のエントリに対するlifetimeをどのように設定するかは、NATルータの性能にも影響する重要な要素であり、NATルータの実装や設計に依存します。

19.2.5 TCP以外のプロトコルの扱い

ここまでは、TCP通信における送信元IPv4アドレスとポート番号、宛先IPv4アドレスとポート番号がそれぞれNATルータでどう変換されるかを見ることにより、一般的なNATの動作を説明しました。実は、TCP以外のプロトコルのNATを介した通信は、この単純な処理では実装できません。

UDPについては、基本的なNATの動作はTCPと同様にUDPのポート番号を利用して実現できます。しかし、TCPにおけるSYN、FIN、RSTのような「通信の開始や終了を明示する仕組み」がUDPにはないので、接続の開始や維持が明示的にできません。そこで実際のNATルータでは、「NATの内側から通信が発生したときにUDPポート番号をもとにNATテーブルのエントリを作成し、一定時間使われなければタイムアウトする」といった実装になっています[†8]。

ICMPについては、通信のフローを示す識別子（TCPやUDPにおけるポート番号に相当するもの）がなく、NATで扱うのが難しいプロトコルです。UDPと同様、通信の開始と終了の把握も困難です。さらに、ICMPパケットが運ぶデータにIPv4アドレスが記述されている場合には、その情報も適切に変更する必要があります。実際のNATルータでは、ICMPパケットのタイプフィールドやメッセージの内容に基づいてICMPパケットに対する変換が実装されています。

19.3 NAT機器に要求される挙動

NATルータなどの機器にどのような挙動が要求されるかについては、RFC 4787[†9]とRFC 5382[†10]、およびRFC 7857[†11]でまとめられています。RFC 4787はユニキャストUDPに対する挙動、RFC 5382はRFC 4787を前提としたTCPに対する挙動、RFC 7857はこれらを更新する内容です[†12]。これら3つのRFCはいずれもBCP（Best Current Practice）とされています。

RFC 4787では、NATの挙動を以下のような項目で説明しています。

- マッピング（後述）
- ポート番号の割り当て
- 変換時のポート番号の偶奇の維持
- 変換時のポート番号の連続性の維持
- マッピングの更新
- 外部IPアドレス空間と内部IPアドレス空間の競合に対する挙動
- フィルタリング（後述）
- ヘアピニング（NAT機器を通じたエンドノード同士の通信）

---

†8　TCPにおいてもFINやRSTなどが送信されずに接続が終了する状況が想定されるので、タイムアウト処理の実装自体はTCPでも必要です。

†9　RFC 4787：F. Audet, C. Jennings, “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP”, 2007年1月

†10　RFC 5382：S. Guha, K. Biswas, B. Ford, S. Sivakumar, P. Srisuresh, “NAT Behavioral Requirements for TCP”, 2008年10月

†11　RFC 7857：R. Penno,S. Perreault,M. Boucadair,S. Sivakumar,K. Naito, “Updates to Network Address Translation (NAT) Behavioral Requirements”, 2016年4月

†12　RFC 4787に対しては、RFC 7857以外にも、CGNに関連する更新がRFC 6888として発行されています。

- ALG（後述）
- 決定論的プロパティ
- ICMP Destination Unreachableの挙動
- フラグメンテーションに関する挙動

そのうえで、NATに対する要求事項を14個定義しています。その要求事項では、ヘアピニングへの対応が必須とされていたり、ポート番号の偶奇の維持が推奨されたりしています（RTPが偶数ポート、RTCPが奇数ポートを利用するという仕様を考慮したもの）。さらに、特殊な状況を除き「短い時間（2分未満）でUDPに対するマッピングを期限切れにしてはならない」という要求や、マッピングやフィルタリングの挙動を途中で変更してはならないという要求もあります。

NAT機器に対する要求仕様の詳細について興味のある方は、RFC 4787をはじめとする関連RFCを参照してください。

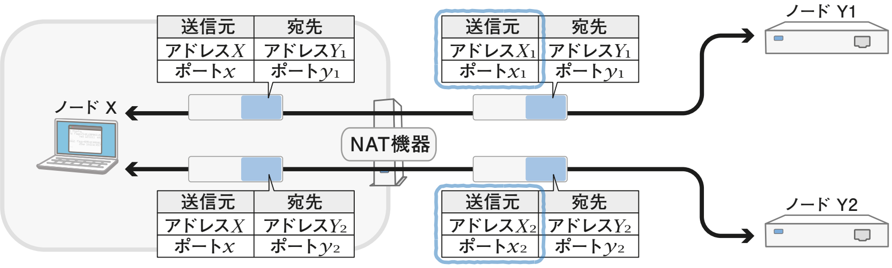
NOTE

NAT越えのためのプロトコルであるSTUNの古い仕様でも、NATを4種類に分類することでNATの挙動が説明されています（第20章参照）。しかし、その分類では実際には不十分であることや、そもそもSTUNでの分類はUDPのみを対象にしたものであることから、現在ではこの分類は推奨されていません。

19.3.1　NAT機器におけるマッピングとフィルタリング

RFC 4787などでは、NAT機器の挙動を**マッピング**および**フィルタリング**という別々の動作に分けて整理しています。たとえば、「家庭内ネットワークからIPv4インターネット宛のパケットに含まれる情報をNATルータでどのように変換するか」は、「マッピングの方法として何を選ぶか」で説明されます。一方、「外部から内部への通信の許可や拒否をどう考えるか」は、「フィルタリングの方法として何を選ぶか」として説明されます。なお、両者はまったく別の動作なので、フィルタリングの方法がマッピングの方法に依存するといったことはありません。

以降では、RFC 4787にならい、図19.4のような構成のネットワークでマッピングの方法とフィルタリングの方法としてどのようなものが考えられるかを説明します（図19.4では、ノードのIPv4アドレスを大文字で、ポート番号を小文字で表しています）。



▶ 図19.4　内部ノードから外部ノードへの2つの通信

■ マッピング

図19.4のNATの内側では、NAT内部のノードXからNAT外部のノードY1およびノードY2への通信における送信元と宛先の情報がそれぞれ次のようになっています。

	X→Y1（NATの内側）	X→Y2（NATの内側）
送信元アドレス	X	X
送信元ポート番号	x	x
宛先アドレス	Y_1	Y_2
宛先ポート番号	y_1	y_2

これらは、途中のNAT機器で、それぞれ次のように変換されています。

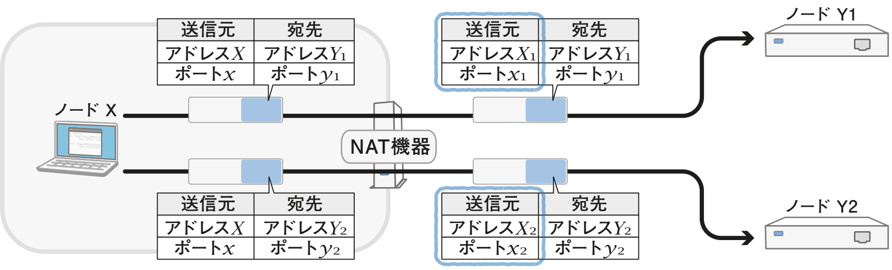
	NAT→Y1（NATの外側）	NAT→Y2（NATの外側）
送信元アドレス	X_1	X_2
送信元ポート番号	x_1	x_2
宛先アドレス	Y_1（同じ）	Y_2（同じ）
宛先ポート番号	y_1（同じ）	y_2（同じ）

このとき、同じノードXからの通信であれば、宛先が違っても、変換後の送信元アドレスや送信元ポート番号を同じにしてよいでしょうか。つまり、$X_1 = X_2$および$x_1 = x_2$としてよいでしょうか。それとも、宛先が別々なので、同じノードXからの通信であってもNATの外側では異なる送信元アドレスやポート番号を使うべきでしょうか。つまり、$X_1 \neq X_2$もしくは$x_1 \neq x_2$とすべきでしょうか。

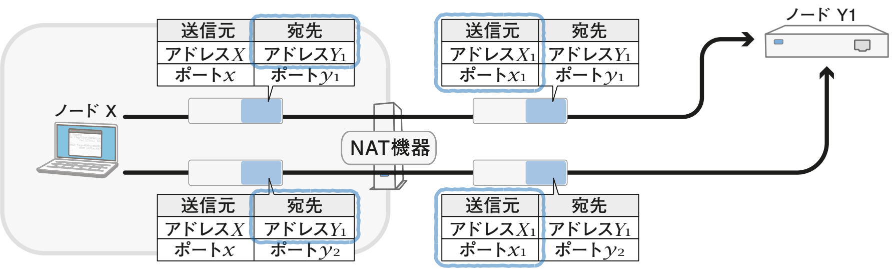
RFC 4787では、こうした「宛先アドレスやポート番号に応じて変換後の送信元アドレスとポート番号を毎回変化させるか、それとも同じ宛先なら毎回同じにするか」の方針をマッピングと呼び、次の3つに分類しています。

- **EIM**（Endpoint-Independent Mapping、エンドポイント非依存マッピング）：NAT内部で同一の送信元からの通信は、宛先によらず、送信元アドレスとポート番号を毎回同じものに変換する（図19.5）
- **ADM**（Address-Dependent Mapping、アドレス依存マッピング）：NAT内部で同一の送信元からの通信は、宛先が同じグローバルIPv4アドレスであれば、送信元アドレスとポート番号を同じものに変換する（図19.6）
- **APDM**（Address and Port-Dependent Mapping、アドレスとポート依存マッピング）：ADMに加えて、さらに宛先ポート番号も同じである場合にのみ、送信元アドレスとポート番号を同じものに変換する（図19.7）

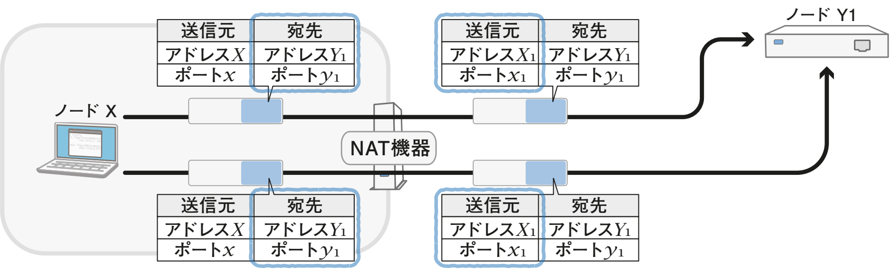
cc BY NC SA



▶ 図 19.5 EIM（エンドポイント非依存マッピング）



▶ 図 19.6 ADM（アドレス依存マッピング）



▶ 図 19.7 APDM（アドレスとポート依存マッピング）

RFC 4787とRFC 5382の大きなポイントのひとつは、これらのうちNAT機器が満たすべき必須の要件をEIMとしていることです。その理由として、EIMであればUNSAF（UNilateral Self-Address Fixing）と呼ばれるプロセスが実行できるようになる点が挙げられます。

UNSAFは、外部からの通信がNAT機器を越えられるようにするために、通信に使うべきアドレスとポート番号を測定するプロセスです。IABによるInformationalなRFCであるRFC 3424[†13]で規定されています。NAT機器を通してリアルタイムメディアやオンラインゲームなどのP2P的な接続を利用できるようにすることが考慮された要件だといえるでしょう。

†13 RFC 3424 : L. Daigle, IAB, “IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation”, 2002年11月

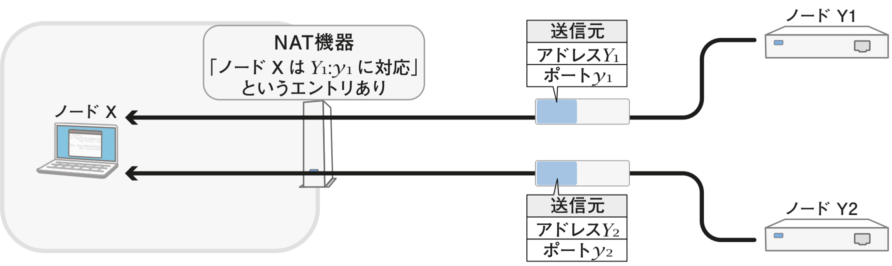
NOTE

RFC 4787では、EIM、ADM、APDMの3種類の挙動を紹介した直後に「3種類の挙動でセキュリティ特性は変わらない」と記されています。セキュリティ特性は、どのようなパケットを通過させるのかによって決まるので、次に説明するNATのフィルタリング挙動によって決まります。

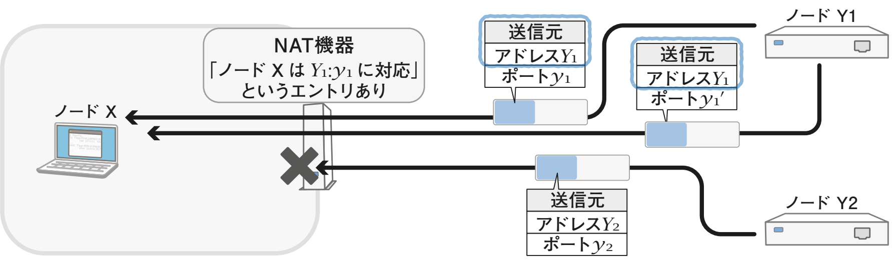
■ フィルタリング

NAT機器では、NAT内部のノードXから、NAT外部のノードYへの通信が新たに開始されると、それらのノード間でアドレスとポート番号についての対応関係が決まります。この対応関係がある状態で、ノードXを宛先とするパケットを外部から受け取ったNAT機器は、それを内部に転送すべきでしょうか。それともフィルタリングすべきでしょうか。RFC 4787では、NAT機器におけるフィルタリングの動作について、次の3種類の基準を示しています。

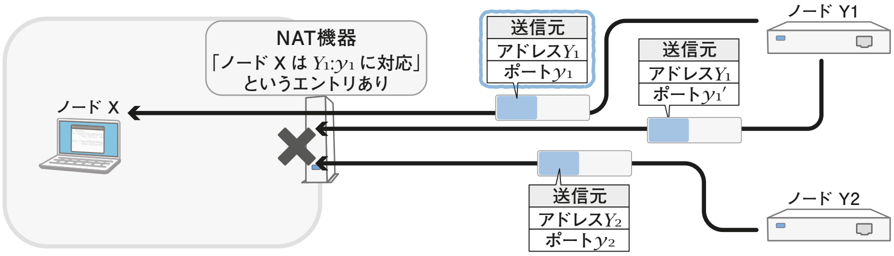
- **EIF**（Endpoint-Independent Filtering、エンドポイント非依存フィルタリング）：NAT外部のノードへの通信がある内部のノード宛のパケットであれば、NAT内部へと転送を許可する（図19.8）
- **ADF**（Address-Dependent Filtering、アドレス依存フィルタリング）：NAT内部からの宛先になったことがあるNAT外部のノードを送信元アドレスとするパケットであれば、NAT内部へと転送を許可する（図19.9）
- **APDF**（Address and Port-Dependent Filtering、アドレスとポート依存フィルタリング）：NAT内部からの宛先になったことがあるNAT外部のノードを送信元アドレスおよびポート番号とするパケットであれば、NAT内部へと転送を許可する（図19.10）



▶ 図19.8　EIF（エンドポイント非依存フィルタリング）



▶ 図19.9　ADF（アドレス依存フィルタリング）



▶ 図19.10 APDF（アドレス依存フィルタリング）

RFC 4787では、通信の透過性が重要である場合にはEIFを推奨しています。

NOTE

NAT機器の挙動は、マッピングとフィルタリングの組み合わせによります。ここで注意が必要なのは、マッピングとフィルタリングは異なる概念だということです。フィルタリングの動作は、マッピングが3種類のいずれであるかとは無関係に決まります。たとえば、マッピングがEIM、フィルタリングがAPDFという実装も考えられます。すでに廃止されたRFC 3489（旧STUN）にあるPort Restricted Coneは、まさにそれに相当しています。

19.3.2 ALG

NATでIPアドレスとポート番号の変換を行うためには、IPヘッダ、TCPヘッダ、UDPヘッダに変更を加える必要があります。しかし、プロトコルによってはさらにペイロードに含まれる内容にも変更が必要です。たとえば、ファイル転送に使われるFTP（RFC 959[†14]）や、IP電話などで使われるSIP（RFC 2543[†15]）では、同時に張られた複数のセッションを識別して既存セッション中で新規セッションの情報を伝えるために、ペイロード中にIPアドレスやポート番号を情報として含むことがあります。

NAT機器でFTPやSIPによる通信を成立させるためには、こうしたプロトコルのIPv4パケットを変換する際に、IPヘッダだけではなくTCPパケットのデータ部分に含まれるメッセージの中身も変換する必要があります。各プロトコルの中身を理解しつつ変換を行う機能はALG（Application Level Gateway）と呼ばれます。

NOTE

実はALGでもまだ通信できない場合もあります。たとえば、IPsecなどで通信が暗号化されていて、その中でIPアドレスなどの情報がやり取りされている場合、ALGでは対応できません。

---

†14 RFC 959：J.Postel and J.Reynolds, “FILE TRANSFER PROTOCOL (FTP)”, 1985年10月

†15 RFC 2543：M.Handley, H.Schulzrinne, E.Schooler, and J.Rosenberg, “SIP: Session Initiation Protocol”, 1999年3月

NATはファイアウォールではない

NATは、内部から外部への通信に応じて動的にフィルタを追加します。この挙動は**SFI**（Stateful Filter Implementation）や**SPF**（Stateful Packet Filtering）と呼ばれます。SFIは簡易なセキュリティとみなされることもあるので、「NATが不要なIPv6ではIPv4より攻撃を受けやすくなる」といわれることもあります。

しかし、NATのSFIはセキュリティを実現する目的で設計されているのではありません。逆に言うと、SFIを実現するためにNATそのものは不要です。そこで、NATを利用せずにIPv6においても「IPv4 NATと同等のセキュリティ」を確保する方法がRFC 4864[†16]で提案されています。IPv6におけるNATの議論については3.12.1項も参照してください。

19.4 IPv4アドレス共有技術の課題

一般のNATには、ISPから割り当てられる1つのIPv4アドレスで複数の機器をインターネットにつなぐという目的があります。家庭や中小企業のネットワークが主な利用環境であり、利用人数も比較的限定されています。それに対し、ISPなどの通信事業者では、グローバルIPv4アドレスの利用数を節約するIPv4アドレス共有技術として大規模なNATを導入することがあります。ここでは、こうしたIPv4アドレス共有技術の課題をまとめます。

19.4.1 CGN（NAT444）とその課題

CGN（Carrier Grade NAT）は、その名のとおりISPのような通信事業者の構内で運用され、多くの契約者が1つの機器へと集約されます。この方法は、次の3種類のIPv4アドレスを利用することから、**NAT444**と呼ばれることがあります。

- ユーザ側のCPE（Customer Premises Equipment）に備わっているNATの内側で利用されるプライベートIPv4アドレス
- CGNで利用する2段めのNATの内側で利用されるプライベートIPv4アドレス
- CGNが接続されているIPv4インターネットで利用されるグローバルIPv4アドレス

NOTE

「CGNという名称ではキャリア（通信事業者）しか使えないようなイメージになってしまうのでLSN（Large Scale NAT）という呼称のほうが適切ではないか」という意見がIETFで強くなったことがあり、LSNという名称が用いられたことがありました。MAN（Multiple Address NAT）という名称が提案されたこともあります。しかし、最終的にはCGNという表現に落ち着きました。

なお、NAT444という用語がIETFで最初に提案されたのは、日本のキャリアやISP数社が共同で2008年に提出した“NAT444 with ISP Shared Address”[†17]というInternet Draftです。

†16 RFC 4864 : G. Van de Velde, T. Hain, R. Droms, B. Carpenter, E. Klein, “Local Network Protection for IPv6”, 2007年5月

(cc) BY NC SA

■ CGNの技術的な特徴

一般的なNAT機器は、1つのプライベートネットワークごとに1つのグローバルIPv4アドレスを利用します。これに対しCGNでは、単一機器が複数のグローバルIPv4アドレスを利用します。扱っているプライベートネットワークの規模が大きいので、1つのプライベートネットワークに対して複数のグローバルIPv4アドレスを使う必要があるのです。

その他、CGNの技術的な側面についてはRFC 6888[†18]で議論されています。ここでは、それらについて説明します。

- **CGNでは複数のグローバルIPv4アドレスを利用する必要がある**

  NATで通信を一意に識別するのは、IPv4ヘッダのプロトコルフィールド、宛先および送信元IPv4アドレス、プロトコルヘッダ情報（プロトコルフィールドがTCPならプロトコルヘッダ中に含まれる宛先および送信元ポート番号）の組み合わせです。そのため、この組み合わせの上限がNATにおける同時通信セッション数の論理的な上限になります。特に、NATが扱える通信セッションの数は、NATで利用するグローバルIPv4アドレスの数によって論理的に制限されてしまいます。

  TCP通信でWebサーバに接続する状況を例に説明しましょう。TCPヘッダのポート番号フィールドは16ビットなので、NATで利用するグローバルIPv4アドレスが1つであった場合、そのNATの内側から特定のWebサーバに対して同時に接続できる数の上限は65536になります[†19]。この論理的な同時接続数の上限を超える大規模なネットワークに導入されるCGNでは、複数のグローバルIPv4アドレスを利用することになります。

  さらにCGNでは、「ユーザごとに同時に通信が可能なセッション数に上限を設定できること」が要求される場合もあります。複数のグローバルIPv4アドレスを多数のユーザで共有して使うことが前提なので、たとえば単一のユーザが大量のTCPセッションを確立してしまうと、他のユーザが使える論理的なTCPセッション数が減ってしまうからです。CGNの物理的なメモリ量などの物理的な制約もあるので、論理的なセッション数の上限以外にも複数のユーザ間で共有が必要な条件が出てきます。

- **同じユーザが同じなるべくIPv4アドレスを利用できるようにする**

  複数のIPv4アドレスをCGNで複数人が共有する場合、同じユーザは同じグローバルIPv4アドレスで通信できるほうが好ましいといえます。もし通信ごとに異なるグローバルIPv4アドレスが使われてしまうと、サーバ側で同じユーザからの複数の通信であることがわかりにくくなってしまうからです。たとえば、WebのためのHTTPはコンテンツごとにTCPセッションを確立することが多いので、TCPセッションごとに毎回異なるグローバルIPv4アドレスが使われてしまうと、同じユーザからの通信を複数のユーザからの通

---

†17 J. Yamaguchi, Y. Shirasaki, A. Nakagawa, H. Ashida, “NAT444 addressing models”, 2012年7月（本書執筆時点の最終的なドラフト）：
https://tools.ietf.org/html/draft-shirasaki-nat444-isp-shared-addr-08

†18 RFC 6888： S. Perreault, Ed., I. Yamagata, S. Miyakawa, A. Nakagawa, H. Ashida, “Common Requirements for Carrier-Grade NATs (CGNs)”, 2013年4月

†19 もしくは、16ビットがすべて0の場合とすべて1の場合を除いた65534個。

cc BY NC SA

信と誤認してしまう可能性があります。そのためCGNでは、複数のグローバルIPv4アドレスを使いつつも、プライベートネットワーク内の同一ユーザに対してはできるだけ同じグローバルIPv4アドレスを割り当てるような仕組みが望ましいとされています。

- 機器に対する特別な要件

一般のNAT機器は、連続運用に耐えることやダウンタイムを限りなくゼロに近づけることよりも、安価であることのほうが優先されがちです。これに対し、CGN機器では、通信事業者での利用に耐えるための性能が求められます。

たとえば、一般的な家庭用NAT機器では、設定を変更するたびにルータの再起動が要求されます。NAT機器に重大なセキュリティホールが存在してファームウェアアップデートが必要となった場合も、ファームウェアアップデート後に再起動が必要になります。しかし、通信事業者が運用するCGNでは、可能な限りダウンタイムを少なくすることが要求されます。CGNでの設定変更やファームウェアアップデートのたびに再起動が必要であっては、通信事業者が顧客に提供するサービスの品質が著しく低下してしまいます。

NAT機器の再起動には、通常のルータの再起動とは異なる面もあります。通常のルータであれば、通信の両端で通信の状態が把握されるので、途中経路におけるルータが一時的に落ちていたとしてもパケットが落ちるだけで済みます。しかし、NAT機器が再起動して変換テーブルが失われると、それまで行われていた通信に関する情報がすべて破棄されてしまうので、それまでの通信がすべてリセットされてしまいます。そこで、再起動による通信のリセットを最小限に抑えるために、変換テーブルを失わずにファームウェアアップデートが可能になる工夫が施されているCGN機器も多くあります。

- ログの保存が必要になる

CGN環境下では、グローバルIPv4アドレスを持つインターネット側からは通信相手を特定することが困難になります。いつ誰がどのような通信を行ったかに関するトレーサビリティを確保するため、CGN機器には変換テーブルをログとして保存する機能が要求されます。

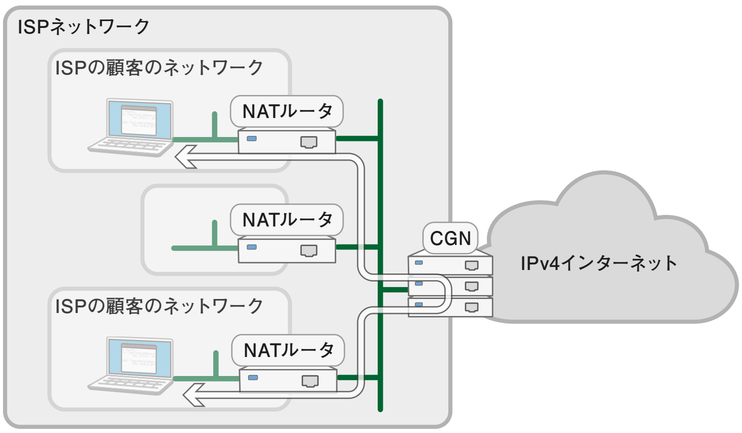
CGNにおけるログの保存は、ISPなどの通信事業者側の視点だけではなく、Webサーバなどの各種サーバ運用者の視点でも重要な問題です。後者については19.4.2項で改めて扱います。

- CGN配下でのユーザ間での折り返し通信

NAT環境では、図19.11のように通信経路がヘアピンのように鋭角に曲がる折り返し通信（ヘアピニング）のサポートが要求される場合があります。ヘアピニングについてはRFC 4787、RFC 5382、RFC 5508[†20]で規定されています。特にCGNでは、P2P利用時にヘアピニングが発生する可能性があるため、対応が必須とされています。

†20 RFC 5508：P. Srisuresh, B. Ford, S. Sivakumar, S. Guha, “NAT Behavioral Requirements for ICMP”, 2009年4月

CC BY NC SA



▶図19.11　ヘアピニング

■ CGNが抱える課題

CGNには、グローバルIPv4アドレスを配布する従来型のインターネット接続サービスと比較してさまざまな課題があります。ここではそのうちのいくつかを紹介します。

CGNが抱える課題のなかには、DS-Lite、MAP-E、NAT64などのIPv4とIPv6共有技術を運用する際にIPv4アドレス共有技術が使われるときも同様に発生するものもあります。

■ P2P的な通信への対応

NATはP2P的な通信を阻害する可能性があります。CGNでは、NATが家庭だけではなくISPなどでも行われることになるので、P2P的な通信がさらに困難になります。

たとえば、多くの家庭用NATルータで実装されているUPnP IGD（Universal Plug and Play Internet Gateway Device）によるNAT越えは、CGN環境下では使えません。UPnPは、その名のとおり、ネットワークに接続された機器を「プラグアンドプレイ」するための機能です。IGDは、UPnPでインターネットに接続されたゲートウェイを制御するための機能です。

UPnPは、マルチキャストで対応機器を発見し、発見した対応機器に対してSOAPを使って情報取得や制御を行います。これは、主にユーザセグメントでの利用を想定された仕様です。UPnPで利用されるマルチキャストアドレスも`239.255.255.250`であり、家庭内にあるNATルータを越えるようなマルチキャストルーティングは考慮されていません。主に同一セグメント内での利用が想定されており、CGNのような環境は想定されていないのです。

UPnPがCGN環境で使えないという背景があったことから、IPv4およびIPv6のNATやファイアウォールに対してポート番号マッピングを行うための仕組みが考案されました。この仕組みはPCP（Port Control Protocol）と呼ばれ、2013年にRFC 6887[†21]として標準化されています。UPnPからPCPを利用するUPnP IGD-PCP IWF（Universal Plug and Play Internet

---

†21 RFC 6887：D. Wing, S. Cheshire, M. Boucadair, R. Penno, P. Selkirk, “Port Control Protocol (PCP)”, 2013年4月

cc BY NC SA

Gateway Device - Port Control Protocol Interworking Function）については、RFC 6970[†22]として標準化されています。PCPは、CGNだけでなく、家庭用NAT、ファイアウォール、NAT64やDS-LiteなどのIPv4/IPv6共存技術でのポート番号マッピングも可能となるように設計されています。

いわゆるNAT越えには、UPnP以外にもさまざまな手法があります。WebRTCなどで利用されるSTUNも、NAT越えのための手法のひとつです。ここでは詳細は割愛しますが、P2P的な通信をNATで実現する具体的な手法に関してはRFC 5128[†23]で紹介されているので参考にしてください。

■ 通信セッションの生存期間

CGN環境には、TCPやUDPなどの通信セッションの生存期間をどれぐらいにするかという課題もあります。通信がまったく行われていないセッションに割かれている資源をいつ解放するかによって、単一の機器で収容できる人数も変わってきます。

インターネットにおける通信セッションは、突然途切れることもあるので、「確実にセッションが切れた」と判断できないことも多くあります。たとえば、Webサイトの閲覧中にいきなりパソコンの通信ケーブルを抜けば、その瞬間のTCP接続は途切れてしまいます。しかし、Webサーバ側にはインターネットの向こう側にいるパソコンの状態はわからないので、通信相手が復帰した場合に備えて、その際のTCP接続の状態が一定時間は維持されます。一定時間が経過すればWebサーバにおけるTCP接続の状態も破棄されますが、途中経路に存在するCGN機器には、WebサーバにおいてTCP接続の状態が破棄された瞬間はわかりません。そのため、何らかのタイミングで「この通信は破棄された」と自分で判断しなければなりません。

一方、TCP接続を切断する際のFINパケットをCGN機器が検知したとしても、セッション数のカウントを残さなければならないような場合もあります。たとえば、TCP接続を切断した側は、切断後に`TIME_WAIT`という状態に入ります。CGN機器では、この状態を、「TCPセッションが維持されている期間」としてカウントしなければなりません（RFC 6269[†24]）。

■ ジオロケーション

IPアドレスは、あまり正確な手法とは言い難いものの、通信相手が物理的に存在している地域の推定に使われることがあります。CGNによって集約される範囲によっては、従来よりも、IPアドレスによる物理的な位置情報の推定誤差が大きくなる可能性があります。

■ ICMPの扱い

ICMPにはTCPやUDPのようなポート番号という概念がないので、NATによる変換を行う際に一意性を確保するための特別対応が必要となります。NATにおけるICMPの扱いについてはRFC 5508で議論されています。特に、MTUが小さすぎてパケットが転送できず、かつフラグ

---

†22　RFC 6970：M. Boucadair, R. Penno, D. Wing, “Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function (IGD-PCP IWF)”, 2013年7月

†23　RFC 5128：P. Srisuresh, B. Ford, D. Kegel, “State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)”, 2008年3月

†24　RFC 6269：M. Ford, M. Boucadair, A. Durand, P. Levis, P. Roberts, “Issues with IP Address Sharing”, 2011年6月

CC BY NC SA

メンテーションができない場合に送信されるICMP Message Too Bigメッセージを CGN機器で転送できないと、Path MTU Discoveryを利用しているアプリケーションが正しく動作せずに通信ができなくなる可能性があります。

なお、ICMPの扱いは、NAT64などのIPv4/IPv6トランスレータでも課題とされています。

■ フラグメンテーション

フラグメンテーションもCGNにおける課題のひとつです。IPv4パケットがフラグメント化されるとき、TCPやUDPなどのトランスポートプロトコルヘッダは、分割されたパケットのうち最初のパケットにのみ含まれます。分割されたパケットのうち2つめ以降にはトランスポートプロトコルのヘッダが含まれないため、CGNにおいて特別な処理が必要となります。

■ IPv4アドレスによるブラックリスト

ネットへの悪質な書き込みや、迷惑メール送信行為に対するブラックリストとして、IPv4アドレスが利用されることがあります。CGN環境では、複数のユーザが同時に同じグローバルIPv4アドレスを利用するので、ブラックリストにそのうちの一部のIPv4アドレスが登録されてしまうと、そのアドレスを共有している無実のユーザにも影響が出る可能性があります。

■ ダイナミックDNSユーザとの相性

自宅にサーバを設置し、外部からアクセス可能にする手法のひとつとして、ダイナミックDNSが利用されることがあります。ISPによってCGNが運用されるようになると、ダイナミックDNSを利用した外部からのアクセスを実現するのが困難になります。

■ ロードバランシング

IPアドレスをもとにロードバランシングを行うような機器では、同じIPv4アドレスを集約したCGNによってトラフィックが上手に分散できなくなってしまう可能性があります。これは、CGNが、極端に多くの通信を行っている特定のホストのように見えてしまうためです。

■ IPv4マルチキャスト

NATにおけるIPv4マルチキャスト対応については、RFC 5135[†25]（BCP 135）で例が紹介されています。本書執筆時点では、グローバルIPv4インターネットにおけるマルチキャストの利用は稀ですが、CGNによってIPv4マルチキャストの実現がさらに困難になる可能性もあります。

■ その他の課題

RFC 6269には、NATなどのIPアドレス共有技術全般における課題がまとめられています。RFC 6269ではCGN、DS-Lite、NAT64、A+Pなどが例として登場しますが、個々の技術ではなく大規模にIPアドレスを共有することに伴う課題そのものが議論されています。

RFC 7021[†26]には、2013年段階のCGNによる通信への影響に対する調査結果がまとめられ

†25 RFC 5135：D. Wing, T. Eckert, “IP Multicast Requirements for a Network Address Translator (NAT) and a Network Address Port Translator (NAPT)”, 2008年2月

†26 RFC 7021：C. Donley, L. Howard, V. Kuarsingh, J. Berg, J. Doshi, “Assessing the Impact of Carrier-Grade NAT on Network Applications”, 2013年9月

cc BY NC SA

ています。

19.4.2 CGNの普及とサーバにおけるアクセスログ

IPv4アドレス在庫が枯渇すると、ISPはこれまでのように「必要になったら新しいIPv4アドレスを申請して割り振りを受ける」ことができなくなります。これは、IPv4アドレスの総量がこれ以上増えなくなるということであり、ISPはそれまで割り振りを受けたIPv4アドレスを節約しながら使い続けなければなりません。その節約の手段としてCGNが活用されることが増えていますが、ISPがCGNを利用するようになると、家庭用NATと合わせて2段階のNATが行われる環境が増えます。多くのユーザが同じIPv4アドレスにまとめられてしまうので、Webサーバ側でプログラムを書いたりサーバを管理したりする側の視点から見ると、ユーザとして見える相手のIPv4アドレスのバリエーションが劇的に減ることになります。

ISPがCGNを利用すると、IPv4アドレスだけでは実際に通信を行った契約回線を特定できなくなるので、サーバ側でアクセスログに記載する項目も変更を迫られることになります。具体的には、TCPのポート番号情報がアクセスログに追加されるようになります。CGN通過後のグローバルIPv4アドレスはまとめられたものであり、契約回線特定のためにはIPv4アドレスとTCPポート番号の両方が必要になるからです。CGNによるIPv4でのインターネット接続サービスが増えるに従い、Webサービスなどのインターネットサービスを提供する側でも、対応が必要になるということです。

RFC 6302[†27]では、インターネットに接続されたサーバでは以下の項目をログとして保存することが推奨されています。

- 送信元ポート番号
- タイムスタンプ
- トランスポートプロトコル（たとえば、UDPやTCP）
- アプリケーションが複数のポート番号を利用する場合には、宛先ポート番号も

サーバのアクセスログについては、NAT機器での時刻情報とサーバログでの時刻情報をどのように同期すればよいかという課題もあります。

19.4.3 ISP Sharedアドレス

CGNから各家庭にあるCPEまでのインターネットサービスプロバイダネットワークにおいて既存のプライベートIPv4アドレスを使ってしまうと、各家庭内で利用されているプライベートIPv4アドレスと競合する可能性があります。そのような競合を起こさずにサービスプロバイダ内ネットワークのグローバルIPv4アドレスを節約できるようにするため、RFC 6598[†28]（BCP153）にて**ISP Sharedアドレス**と呼ばれる下記のアドレスブロックが確保されました。

---

†27　RFC 6302：A. Durand, I. Gashinsky, D. Lee, S. Sheppard, “Logging Recommendations for Internet-Facing Servers”, 2011年6月

†28　RFC 6598：J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, M. Azinger, “IANA-Reserved IPv4 Prefix for Shared Address Space”, 2012年4月

cc BY NC SA

- 100.64.0.0/10

このアドレスブロックはARINから出されたもので、2012年3月時点で確保されており、whoisには「SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED」として登録されています[†29]。ISP Sharedアドレスは、本書執筆時点において、すでにさまざまなISPによって活用されています。

ISP Sharedアドレスにより、10.0.0.0/8、172.16.0.0/12、192.168.0.0/16に続く4つめのプライベートIPv4アドレスが誕生したとも考えられます。ただし、100.64.0.0/10はサービスプロバイダネットワーク内でのみ利用される点で、他の3種類のプライベートIPv4アドレスとは異なります。

†29 Whois-RWS SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED ：https://whois.arin.net/rest/net/NET-100-64-0-0-1

[書籍版では空白のページです]

第20章

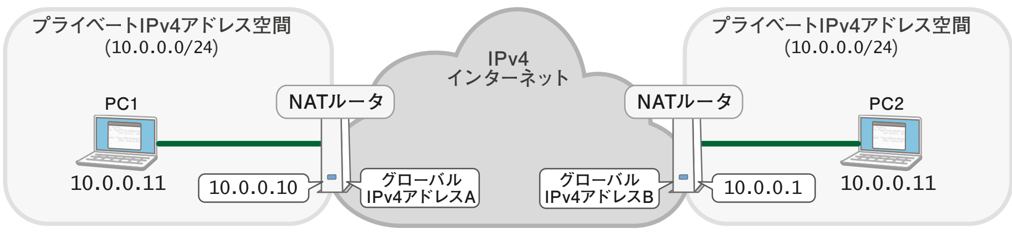
STUNとTURN

NATの内側に設置された機器には、「インターネット側から自分がどんなグローバルIPv4アドレスなどで見えているか」がわかりません。そのため、P2P的な通信を行うために必要な情報を収集する手法として、STUN（Session Traversal Utilities for NAT）と呼ばれる仕組みが開発されました。さらに、そもそもNATの外側と直接通信できない場合に中継ノードを介して通信を実現する仕組みとして、TURN（Traversal Using Relays around NAT）と呼ばれる仕組みもあります。

本章では、STUNおよびTURNについて説明します。これらはIPv4アドレス在庫枯渇問題に直接関係して登場した技術ではありませんが、いわゆる「NAT越え」を実現する仕組みであることから、NATに関連する要素技術として理解が必要になる場面があります。

20.1 STUN

例として図20.1のような状況について考えます。この図では、PC 1とPC 2はそれぞれ別のプライベートなネットワークに接続されており、両方とも自分自身のIPv4アドレスを`10.0.0.10`としています。PC 1もPC Bも、NATルータの外側で自分の通信にどんなグローバルIPv4アドレスが使われているかを知らないので、相手のIPv4アドレスを指定してP2Pの通信を開始することはできません。

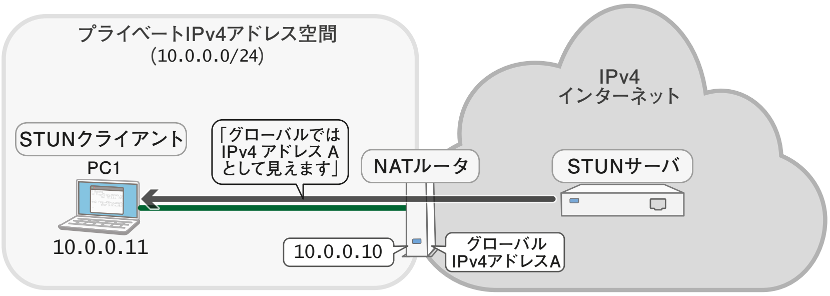


▶ 図20.1　NATで運用されたネットワークに接続された機器同士

STUNでは、この状況を解決するために、グローバルIPv4アドレスを持つSTUNサーバを利用します。NATルータの内側で運用されているPCは、「自分の通信で利用されているグロー

cc by nc sa

バル IPv4 アドレス」などの情報を、STUN サーバから受け取ります。

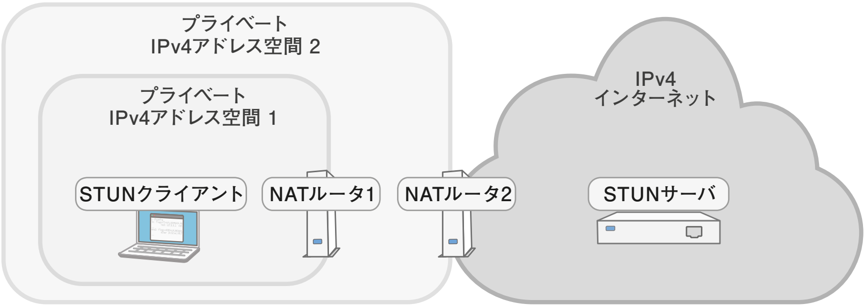


▶ 図 20.2　STUN によるグローバル IPv4 アドレスの把握

NOTE

実際の通信が STUN サーバのみで完結するとは限らず、何らかのマッチングを行う別のサーバも必要になることが多いという点には注意が必要です。

STUN は、RFC 8489[†1] で定義されています。RFC 8489 には、STUN の利用環境の例として、図 20.3 のような 2 段の NAT を有する環境が紹介されています。



▶ 図 20.3　RFC 8489 における STUN の想定環境例

図 20.3 のような例が RFC で紹介されていることから、STUN のユースケースは CGN の存在を強く意識したものであることがわかります。ただし、STUN そのものは 2 段 NAT 環境に限定された仕組みではありません。一般的な SOHO ルータにおける NAT のみの 1 段 NAT 環境でも利用可能なプロトコルです。

20.1.1　旧 STUN と新 STUN

STUN には、無数にある NAT 機器の実装に依存する部分が多くあります。そのため、実運用の中で得られた経験から、仕様がいったん大きく見直されました。さらに、その仕様見直しによって、かつては非常に重要な考え方とされた「NAT の分類」まで変わってしまいました。そういった意味では、多少ややこしいプロトコルであるともいえます。

---

†1　RFC 8489：M. Petit-Huguenin,G. Salgueiro,J. Rosenberg,D. Wing,R. Mahy,P. Matthews, “Session Traversal Utilities for NAT (STUN)”, 2020 年 2 月

CC BY NC SA

STUNは、最初はRFC 3489[†2]として定義されました。その後、2008年にRFC 3489を上書きして廃止するRFC 5389[†3]が発行され、旧STUNを利用することは推奨されなくなりました。さらに2020年には、RFC 5389がRFC 8489により廃止されています。

同じSTUNという呼称であることから、同一のプロトコルであると誤解されがちですが、RFC 8489はRFC 3489とは根本的にまったく異なるプロトコルを規定していると考えたほうがよいでしょう。そもそも、実は名前もまったく違います。旧STUNの名称は、“Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”を短縮したものとされていました。これに対し、新STUNの正式な名称は“Session Traversal Utilities for NAT”です。旧STUNは、名称からわかるように、UDPを前提としたものでした。新STUNはメッセージをTCPやTLSでもやり取りできるように変更されています。

新STUNで想定されている利用環境は、1つもしくは複数のNAT機器の裏側にクライアントが存在しているような環境です。STUNクライアントからのリクエストに対して、STUNサーバがレスポンスを送信します。これらの基本的な構造は旧STUNも同様ですが、旧STUNと新STUNではメッセージの種類が異なります。旧STUNでは、クライアントとサーバ間で、Binding RequestとShared Secretという2種類のメッセージを利用していました。新STUNでは、これがBindingという1種類のメッセージだけになっています。

新STUNは、NAT越えの機能を実現するためのツールの一部という位置づけであり、実際の用途はそれぞれの仕組みに依存します。20.3節で後述するTURNもそのような仕組みのひとつです。その他、新STUNを定義しているRFC 8489では、新STUNと連携してNAT越えを実現する仕組みとしてICE（RFC 8445[†4]）やSIP Outbound（RFC 5626[†5]）が挙げられています。

20.2 旧STUNにおけるNATの分類

旧STUNを定義しているRFC 3489では、NATを下記の4種類に分類し、これらのNATの種別を検出するための機能も規定されていました。

- Full Cone NAT
- Restricted Cone NAT
- Port-restricted Cone NAT
- Symmetric NAT

その後、これらをNATの分類として利用することは推奨されなくなりました。その代わり、

---

†2 RFC 3489：J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, 2003年3月

†3 RFC 5389：J. Rosenberg, R. Mahy, P. Matthews, D. Wing, “Session Traversal Utilities for NAT (STUN)”, 2008年10月

†4 RFC 8445：A. Keranen,C. Holmberg,J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal”, 2018年7月

†5 RFC 5626：C. Jennings,R. Mahy,F. Audet, “Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)”, 2009年10月

NATにおけるユニキャストUDPの挙動に関する要求事項をまとめたRFC 4787[†6]が参照されるようになりました。RFC 4787では、RFC 3489のようにNATを4種類に分けるのではなく、マッピングやフィルタリングといった動作の組み合わせとしてNAT機器の挙動を説明しています（19.3節）。新STUNのRFCでも、旧STUNにおけるNATの分類方法は含まれておらず、RFC 4787が参照されています。

NOTE

RFC 4787では、「RFC 3489では、Full Cone、Restricted Cone、Port-restricted Cone、Symmetricという表現が利用されているが、それらの分類は実際のNATの挙動を示すには不適切であることが証明されており、多くの混乱の原因となった」としています。

とはいえ、旧STUNの定義で利用されているFull Cone NATなどの表現は、後述するTeredoの仕様などで今でも使われることがあります（22.2節参照）。そのため、ここでは旧STUNにおけるNATの4分類のそれぞれについて簡単に解説しておきます。

20.2.1　Full Cone NAT

Full Cone NATでは、特定の内部IPアドレスとポート番号の組を、特定の外部IPアドレスとポート番号にマッピングします。NAT外部の特定のアドレスとポート宛の通信が、送信元IPアドレスに依存せず、NAT内部の単一のIPアドレスとUDPポートに転送されるような構成です。Full Cone NATでは、NAT機器においてセッションが確立すると、複数のIPアドレスから特定の内部IPアドレスにUDPパケットを送信可能になります。

図20.4の例では、NAT内から`203.0.113.3`のUDP 2222番ポートに向けてUDPパケットが送信されています。NATの外側では、このUDPパケットの送信元IPアドレスは`198.51.100.2`、UDP送信元ポートは1111となっています。このとき、NATの外側から`198.51.100.2`のUDPポート1111に向けてインターネット上の他のホストからUDPパケットが送信されると、そのUDPパケットはNAT機器を越えてNATの内側の`192.168.0.2`に転送されます。

20.2.2　Restricted Cone NAT

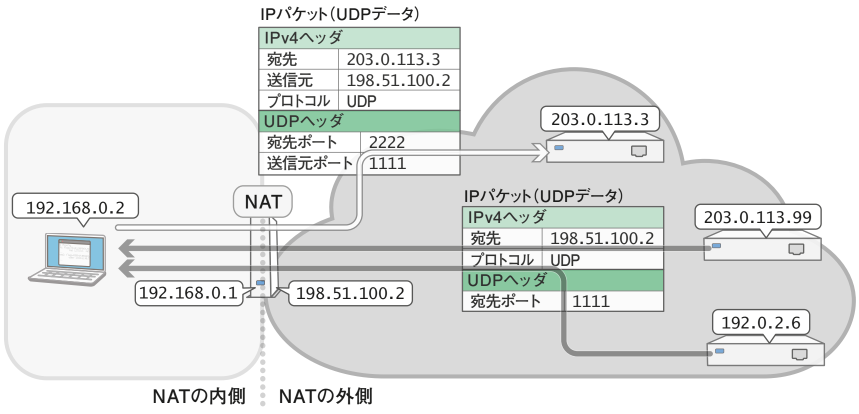
Restricted Cone NATでは、Full Cone NAT同様に、特定の内部IPアドレスとポート番号の組が特定の外部IPアドレスとポート番号の組へとマッピングされます。Full Cone NATとの違いは、外部から内部へパケットが届くために、内部からその外部ホストに対してパケットの送信が過去に行われている必要があるという点です。

図20.5の例では、NAT内部からUDPパケットが送信されたことがある`203.0.113.3`および`203.0.113.99`からのUDPパケットはNAT内部へと転送されますが、NAT内部からUDPパケットを受け取っていない`192.0.2.6`からのUDPパケットはNAT内部へと転送されずに破棄されています。

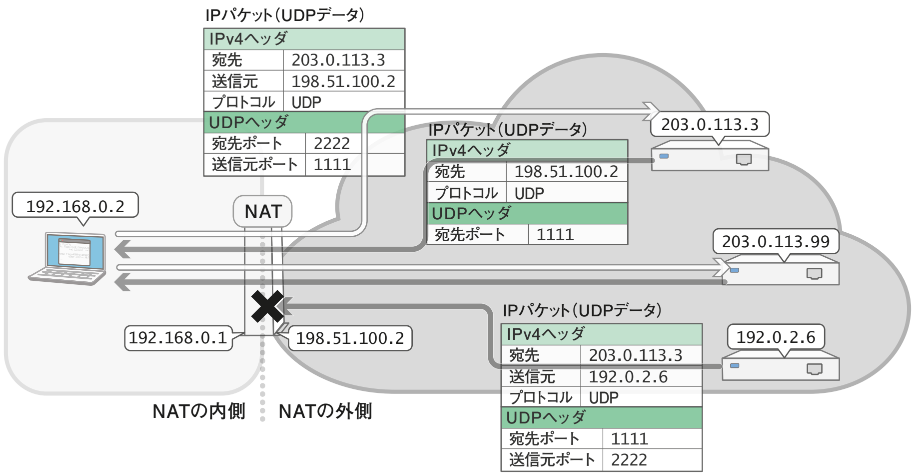
---

†6　RFC 4787：F. Audet, C. Jennings, “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP”, 2007年1月

(cc) BY NC SA



▶ 図 20.4　Full Cone NAT



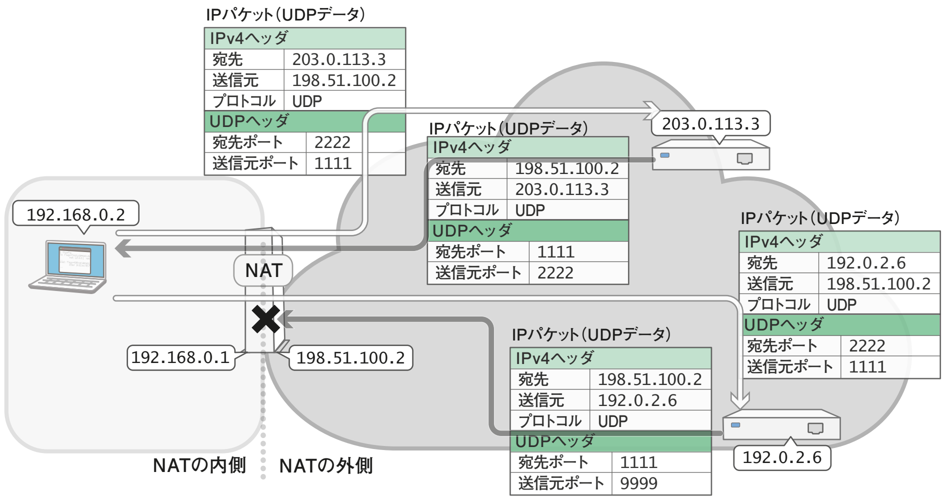
▶ 図 20.5　Restricted Cone NAT

20.2.3　Port-restricted Cone NAT

Port-restricted Cone NAT は、Restricted Cone NAT に対してポート番号による制限を加えたものです。NAT 内部からパケットを送信したことがある外部 IP アドレスとポート番号からのパケットのみが、NAT 内部へと転送されます。

図 20.6 の例では、NAT 内部から受け取った UDP パケットの送信元ポートを宛先ポートとして NAT 外部から送信される UDP パケットは、NAT 内部へと転送されます。一方、この例では `192.0.2.6` は NAT 内部から UDP パケットを受け取っていますが、`192.0.2.6` からの UDP パケットの送信元 UDP ポートが NAT 内部からの UDP 宛先ポートとは異なる 9999 となっているので、これは NAT 内部へと転送されずに破棄されています。

cc BY NC SA

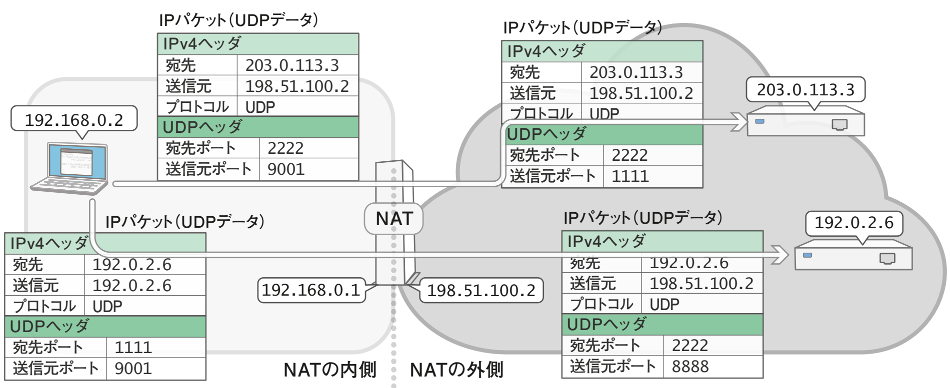


▶ 図20.6　Port Restricted NAT

20.2.4　Symmetric NAT

Symmetric NATでは、内部IPアドレスとポート番号の組と外部IPアドレスとポート番号の組が個別に扱われます。送信元パケットが利用する内部IPアドレスとポート番号の組が同じであっても、別IPアドレスに対するセッションに対しては別の外部ポート番号が割り当てられます。外部からのパケットは、内部から送信されたことがある外部IPアドレスおよびポート番号からのもののみが内部へと転送されます。

図20.7の例では、NAT内部からのUDPセッションは送信元と宛先IPアドレス、送信元と宛先UDPポートの4つの組ごとに個別に扱われます。NAT内部での送信元UDPポート番号は`203.0.113.3`と`192.0.2.6`の両方に対して9001となっていますが、インターネットではNAT機器によって別々の送信元UDPポートとなって送信されます。



▶ 図20.7　Symmetric NAT

cc BY NC SA

20.3 TURN

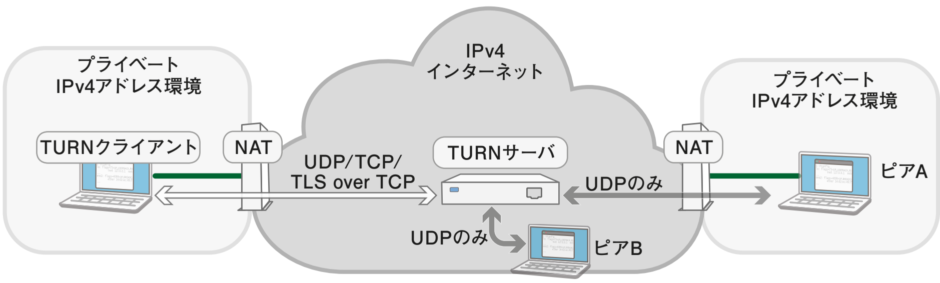
NATルータの挙動によっては、外部との直接通信ができない場合があります。そのような場合には、通信を仲介するノードの手助けが必要になります。その仲介のためのプロトコルが、RFC 5766で定義されているTURNです。

NOTE

RFC 5766[†7]で定義されているTURNはIPv4のみに対応していますが、2011年に発行されたRFC 6156[†8]によってIPv6におけるTURNの利用も可能になっています。

RFC 5766は、“Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”と題されていることからわかるとおり、STUNに対する拡張として定義されています。実際、TURNで使われるフォーマットの多くは、STUNと類似したものです。

TURNの動作環境例を図20.8に示します。STUNによるNAT越えが困難な環境に対応するための中継サーバとなるのが、TURNサーバです。TURNクライアントは、NATの内側のプライベートIPv4アドレス環境に接続されており、TURNサーバを経由して「ピア（peer）」と呼ばれる他のノードと通信が可能になります。



▶図20.8　TURN概要

TURNでは、NAT環境下のTURNクライアントからTURNサーバへの通信手段として、UDP、TCP、TLS over TCPの3種類の通信が利用可能です。一方で、TURNサーバからピアへの通信としては、UDPのみが採用されています。これは、NAT環境から外への通信として、TCPやTLS over TCPのみがファイアウォールなどで許可されている環境に対応するためです。

TURNクライアントからTURNサーバへの通信	TURNサーバからピアへの通信
UDP	UDP
TCP	UDP
TLS over TCP	UDP

†7 RFC 5766：R. Mahy,P. Matthews,J. Rosenberg, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”, 2010年4月

†8 RFC 6156：G. Camarillo,O. Novo,S. Perreault, “Traversal Using Relays around NAT (TURN) Extension for IPv6”, 2011年4月

CC BY NC SA

TURNサーバを経由してピアと通信するTURNクライアントは、何らかの方法によって、ピアとの通信に必要なピアのIPアドレスやポート番号を知る必要があります。しかし、その方法はTURNプロトコルの範疇外です。RFC 5766では、電子メール、ICE、SIPなどの方法が利用可能であることが記載されています。

20.3.1　TURNサーバの自動的な発見

Standards TrackのRFCであるRFC 8155[†9]では、TURNサーバを自動的に発見する方法が定義されています。RFC 8155で定義されているのは、DNSのS-NAPTR（Straightforward-Naming Authority Pointer）リソースレコードを利用する方式、DNS-SD（DNS-based Service Discovery、RFC 6763[†10]）とmDNS（Multicast DNS、RFC 6762[†11]）を利用した方式、IPエニーキャストを利用する方式の3つです。IPエニーキャストによるTURNサーバ発見のために、`192.0.0.10/32`と`2001:1::2/128`が予約されています。

†9　RFC 8155：P. Patil,T. Reddy,D. Wing, “Traversal Using Relays around NAT (TURN) Server Auto Discovery”, 2017年4月

†10　RFC 6763：S. Cheshire,M. Krochmal, “DNS-Based Service Discovery”, 2013年2月

†11　RFC 6762：S. Cheshire,M. Krochmal, “Multicast DNS”, 2013年2月

第V部

IPv4/IPv6共存技術

IPv6への対応が進んでも、インターネットではNATとともにIPv4も引き続き利用されています。そうしたIPv4とIPv6をつなぐためのIPv4/IPv6共存技術として、ここでは次のようなものを紹介していきます。

- 6to4
- Teredo
- ISATAP
- 6rd、4rd
- 6PE
- NAT64 + DNS64
- DS-Lite
- lw4to6
- A+P
- MAP-E、MAP-T
- 464XLAT

IPv4/IPv6共存技術に関しては、IPv6の基本仕様が策定されてからというもの、活発な議論が続いています。当初は「IPv4からIPv6へとユーザを移行させる」という意識が強かったこともあり、IPv6移行技術（IPv6 Transition Mechanism）と呼ばれていたこともありました。しかし、2010年頃からは、IPv4/IPv6共存技術（IPv4 IPv6 Co-existence）という表現が利用されることが増えました。

[書籍版では空白のページです]

cc BY NC SA

第21章

IPv4/IPv6共存技術の分類

IPv4/IPv6共存技術にはたくさんの種類があります。「たくさんのIPv4/IPv6共存技術があって煩雑だ」という感想を持っている方も少なくないでしょう。

IPv4/IPv6共存技術が煩雑に見える背景には、IPv6がIPv4とはまったく異なるプロトコルであることだけでなく、インターネットそのものがさまざまなバリエーションの技術によって運用されており、ネットワーク構成も大きく異なる場合が多いという側面があります。従来どおりの品質でIPv4インターネットへの接続性を提供しつつ、IPv6インターネットへの接続性を実現することには、さまざまなネットワークに固有の難しさがあります。IPv4/IPv6共存技術を理解するには、「このIPv4/IPv6共存技術はどういった環境を想定して作られたものか」という視点で、仕様の設計思想に着目してみてください。たくさんのIPv4/IPv6共存技術があっても、各事業者が利用を検討するときには、選択肢は自ずと1つか2つに絞られるはずです。

ネットワーク事業者にとって、ネットワーク構成の違いはビジネス構造の違いでもあります。その意味で、IPv4/IPv6共存技術の標準化動向は、ビジネスの方向性を左右するものだといえるでしょう。

21.1 IPv4/IPv6共存技術のバリエーション

本書では、説明のため、IPv4/IPv6共存技術を以下のように分類します。

- トンネル技術
- 変換技術
- 組み合わせによる運用手法

トンネル技術は、IPトンネルを構築することによって通信路を確保するための技術です。本書で扱うトンネル技術は、IPv4トンネルでIPv6パケットを運んだり、逆にIPv6トンネルでIPv4パケットを運ぶようなものです。たとえば、IPv6接続性が確保できずにIPv4接続性だけが確保できる環境で、IPv4を利用してトンネルを構築し、IPv6が接続できる場所までIPv6パケットを運ぶような技術があります。IPv4でIPv6パケットを運ぶトンネル技術の主な目的は、接続性の確保です。一方、IPv6でIPv4パケットを運ぶ技術は、IPv6を利用して基幹ネットワー

CC BY NC SA

クを構築することで運用コストを下げることを目指しています。両者で想定しているメリットが異なる点に注意してください。

変換技術は、IPv4による通信をIPv6に変換したり、逆にIPv6による通信をIPv4に変換したりするものです。変換技術を利用することで、エンドノードは、自らが対応していないバージョンのIPプロトコルで運営された相手と通信が可能になります。

組み合わせによる運用手法は、トンネル技術や運用技術などを組み合わせることを指しています。各種の技術を組み合わせることによって、IPv4とIPv6の両方に対応しやすくすることを意図したものです。

21.2 ステートフルとステートレス

IPv4/IPv6共存技術の仕組みを理解するとき、そのプロトコルがステートフルであるか、ステートレスであるかは非常に大きな要素です。以降で紹介する各種プロトコルの説明を読むとき、そのプロトコルがステートフルであるかステートレスであるかに注目すると、プロトコルの設計思想や制限事項が垣間見えてくると思います。

ステートフルなプロトコルは、各通信セッションの状態をルータなどが保持する必要があります。そのため、規模性が制限されると同時に、実装コストが高くなりがちです。

ステートレスなプロトコルは、各通信セッションの状態に依存せずに処理ができます。あらかじめ決められた決まり事に応じて処理を行うだけなので、ステートフルなプロトコルと比べると規模性があり、実装コストも低くなります。しかし、ステートフルなプロトコルと比べると細かい制御ができない傾向があります。

21.3 IPv4/IPv6共存技術利用のパターン

IPv4とIPv6の混在環境をどのように構築するかには、いくつかのパターンがあります。それらを紹介しているRFCとして、RFC 4213[†1]やRFC 6180[†2]などが挙げられます。

以下に、考えられる4つのパターンを要約します。

- IPv4とIPv6のデュアルスタック環境を構築する
- IPv4だけで構成されているネットワークを経由してIPv6インターネットへと接続する（IPv4の島を越えてIPv6へと接続するイメージ）
- コアネットワークをIPv6だけで構築することにより、コアネットワークのデュアルスタック運用を避け、運用コストを軽減する
- IPv6のみでの運用を基本とするネットワークを構築する。IPv4インターネットとの通信が必要な場合には、IPv6ネットワークを経由してIPv4インターネット上のホストと通信を行える仕組みを用意する

---

†1 RFC 4213：E. Nordmark, R. Gilligan, “Basic Transition Mechanisms for IPv6 Hosts and Routers”, 2005年10月

†2 RFC 6180：J. Arkko, F. Baker, “Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment”, 2011年5月

CC BY NC SA

これらのパターンに応じて、環境によってどのようなIPv4/IPv6共存技術が採用されるのかが変わります。たとえば、IPv4だけで構成されているネットワークを経由してIPv6インターネットへと接続するために採用される主なIPv4/IPv6共存技術は、トンネル技術になるでしょう。

これまでに提案されてきたさまざまなIPv4/IPv6共存技術には、時代背景による差もあります。IPv4アドレスの中央在庫が枯渇するまでは、IPv6によるインターネット接続サービスが稀だったので、IPv4ネットワークを経由してIPv6インターネットへの接続を実現するIPv4/IPv6共存技術の提案が多く見られました。その後は、IPv6ネットワークの普及を前提として考慮されたIPv4/IPv6共存技術の提案が増えてきました。

IPv6の普及が進み始めたこともあり、IPv4をIPv6ネットワーク上で仮想的に提供するIPv4aaS（IPv4 as a Service）という考え方も登場しています。これは、クラウドの世界で使われているSaaS（Software as a Service）、PaaS（Platform as a Service）、IaaS（Infrastructure as a Service）のように、IPv4をサービスのひとつとして提供するというイメージでの表現です。IPv6の普及が進めば進むほど、そういった考え方での設計や運用が増えそうです。IPv4aaSに関しては、RFC 8585[†3]として、CE（Customer Edge）ルータに要求される機能をまとめたInformationalなRFCが2019年に発行されています。

IPv4/IPv6共存技術の調査検討が必要という場合、IPv6対応ネットワークを構築しつつ純粋なIPv4とIPv6のデュアルスタック環境を構築するのが難しい状況が多いでしょう。その際には、どのような環境でIPv4/IPv6共存技術を利用するかはもちろん、業界動向を含めた全体的な流れを調査することも大切です。

21.4 今後も多くの試みが誕生する

IPv4とIPv6が共存する状態はしばらく続くことが予想されるので、IPv4/IPv6共存技術に関する提案や取り組みは今後も続くでしょう。議論や実運用を通して問題点が明らかになることで廃止される手法もいろいろありそうです。

IPv4/IPv6共存技術について検討する際に重要なのは、自分の環境に適合した技術がどれであるかを理解することです。たくさんの技術があるように見えるのは、インターネットがさまざまな環境で運営されているためであり、そのすべての環境で利用可能な万能の技術はありません。言い換えれば、必ずしもすべての手法を理解する必要もないのです。

これからIPv4アドレス在庫枯渇に伴う問題が深刻化するとともに、IPv6の普及が進めば、数多くのIPv4/IPv6共存技術が開発され運用されていくと考えられます。

---

†3 RFC 8585 : J. Palet Martinez, H. M.-H. Liu, M. Kawashima, “Requirements for IPv6 Customer Edge Routers to Support IPv4-as-a-Service”, 2019年5月

CC BY NC SA

[書籍版では空白のページです]

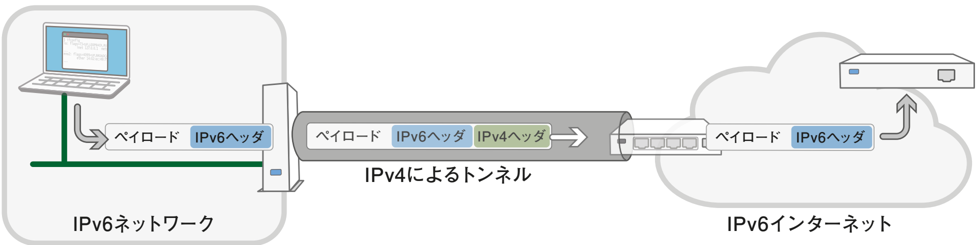
CC BY NC SA

第22章

トンネル技術

IPv4/IPv6共存技術のなかには、パケットをパケットでカプセル化するトンネル技術を利用しているものが数多くあります。IPv4/IPv6共存技術におけるカプセル化には、IPv6パケットをIPv4パケットでカプセル化する場合と、IPv4パケットをIPv6パケットでカプセル化する場合があります。

IPv6パケットをIPv4パケットで**カプセル化**するトンネル技術（図22.1）は、IPv6インターネットへの接続性がなくIPv4インターネットへの接続性しかない環境において、IPv6インターネットを利用する目的で利用されます。IPv4ネットワークを利用したIPv6パケット転送の仕組みについては、RFC 4213[†1]で議論されています。



▶ 図22.1　IPv4によるトンネル

IPv4ネットワークでIPv6パケットを運ぶのとは逆に、IPv6ネットワークでIPv4パケットを運ぶトンネル技術もあります。RFC 2473[†2]では、IPv6を利用した各種プロトコルについて、パケットのカプセル化が定義されています。RFC 2473は1998年のRFCということもあり、記載されているプロトコルにはIPv4のほかAppleTalk、IPX、CLNPなども含まれています。

IPv6ネットワークを利用したIPv4パケットの転送の主な目的は、IPv6のシングルスタックネットワークからIPv4インターネットへの接続性を提供することです。将来におけるIPv4とIPv6のデュアルスタック運用に伴う負荷の軽減や、IPv4アドレス節約を意識した技術といえ

†1　RFC 4213：E. Nordmark, R. Gilligan, “Basic Transition Mechanisms for IPv6 Hosts and Routers”, 2005年10月

†2　RFC 2473：A. Conta, S. Deering, “Generic Packet Tunneling in IPv6 Specification”, 1998年12月

CC BY NC SA

るでしょう。

IPv6パケットをIPv4パケットでカプセル化する場合と、IPv4パケットをIPv6パケットでカプセル化する場合とでは、利用する技術が異なるだけでなく、トンネルを張る主体やユースケースも異なります。IPv6パケットをIPv4パケットでカプセル化する場合、トンネルを張るのは一般ユーザであると想定されます。IPv6サービスを提供していないISPなどで、「IPv6インターネットへの接続性を確保するために一般ユーザがトンネルを張る」というユースケースです。一方、IPv4パケットをIPv6パケットでカプセル化する場合のユースケースは、「コスト削減や運用性向上のため、ISPなどのネットワーク管理者がトンネル技術を利用する」というものです。たとえば、24.1節で紹介するDS-Liteは、IPv6トンネルでIPv4を転送することで、ネットワークのコア部分をIPv6のみで実現できる設計になっています。

IPv4/IPv6共存のためのトンネル技術にはセキュリティ上のリスクもあります。トンネル技術そのものが抱えるセキュリティリスクに関しては22.7節で紹介します。また14.10節では、IPv4ネットワークにおけるIPv6の利用で考えられるセキュリティ上の課題について紹介しています。

22.1　6to4

6to4は、IPv4/IPv6共存を目的としたトンネル技術のための基本的な仕組みとして、RFC 3056[†3]で規定されたプロトコルです。IPv4での接続サービスしかないユーザがIPv6インターネットへの接続性を得られるように、IPv6パケットをIPv4トンネルを通じてIPv6インターネットへと届ける仕組みです。Windows VistaやWindows 7、Mac OSX、Linux、各種BSD系OSなど、さまざまなOSが6to4に対応していました。6to4機能を持つSOHOルータなども世界中で稼働しています。たとえば、2017年10月に行われたNANOG 68の発表でも、6to4によるトラフィックが存在していることが示されています[†4]。

ただし、6to4は2001年に標準化された古いプロトコルである点に注意が必要です。かつては、多くのエンドユーザがISPなどの直接的な協力を得ずにIPv6環境を手軽に試せる仕組みとして重宝されましたが、さまざまな欠点も抱えていることから、本書が出版された時点では利用が推奨されない古いプロトコルとなっています。たとえば、2012年9月に発行されたRFC 6724[†5]では、6to4によるIPv6よりも、IPv4での接続を優先するデフォルト設定を規定しています。

現在では利用が推奨されていない6to4ですが、その考え方や、それが廃止へと向かった経緯を理解することは、IPv4/IPv6共存技術の考え方を知るうえで重要です。そこで本書では6to4について少し詳しく解説します。6to4の問題点については22.1.4項で説明します。

---

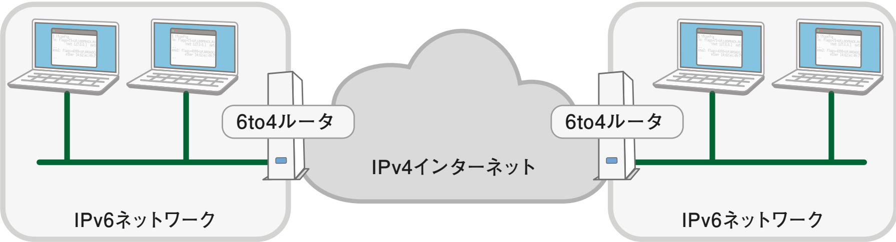
†3　RFC 3056：B. Carpenter, K. Moore, “Connection of IPv6 Domains via IPv4 Clouds”, 2001年2月

†4　NANOG Meeting Presentation Abstract ： `https://nanog.org/meetings/abstract?id=2948`

†5　RFC 6724：D. Thaler, R. Draves, A. Matsumoto, T. Chown, “Default Address Selection for Internet Protocol Version 6 (IPv6)”, 2012年9月

22.1.1 6to4の仕組み

6to4では、IPv4パケットでIPv6パケットをカプセル化して運びます。図22.2のような互いに独立したIPv6ネットワークを、IPv4インターネットでつなぐために考案された仕組みです。なお、図22.2は、IPv6インターネットが世界的に普及しておらずIPv6ネットワークが局所的に点在していた状況を想定したものです。



▶ 図22.2 6to4概要

図22.2では、IPv6パケットが6to4ルータによってIPv4カプセル化され、IPv6インターネットとの仲介を行う6to4ルータへとIPv4インターネットを通じて転送されます。IPv4でカプセル化されたIPv6パケットを受け取った6to4ルータは、IPv6パケットをIPv4パケットから取り出し、それをIPv6ネットワークへと転送します。

6to4をIPv6インターネットとの接続に利用することも可能です。このとき6to4では、6to4ルータから送信されるIPv6パケットの送信元IPv6アドレス中に、IPv6インターネットとのやり取りを担う6to4リレールータ（次項で説明）経由で受け取ったIPv4パケットのIPv4送信元情報を埋め込みます。IPv6アドレスがIPv4アドレスよりも長いことを利用して、6to4リレールータへの負荷を減らすために、6to4リレールータにおいて状態を保持しなくても済むように設計されているのです。

6to4で利用されるIPv6アドレスは`2002:V4ADDR::/48`です（図22.3）。V4ADDRの部分には、6to4ルータのグローバルIPv4アドレスが埋め込まれます。V4ADDRに続くSLA IDとインターフェース識別子の部分には、必要に応じて適切な値が代入されます。



▶ 図22.3 6to4におけるIPv6アドレス

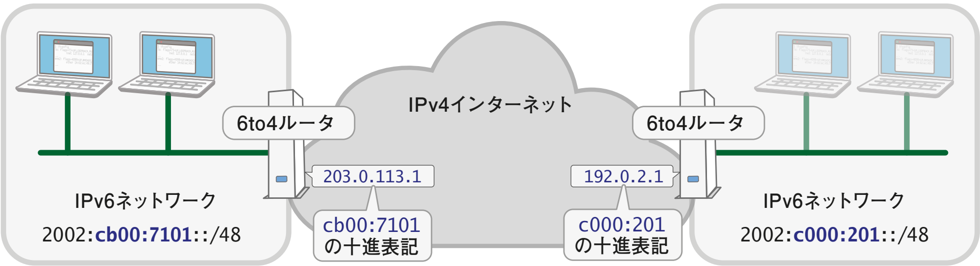
6to4でIPv6パケットをカプセル化するIPv4ヘッダでは、プロトコルIDフィールドとして、IPv6を表す41が指定されます。

22.1.2 6to4ルータと6to4リレールータ

6to4の仕組みそのものは上記のように非常に単純です。ただし、実際の通信を考えるともう少し複雑で、「6to4サイト同士の通信」と「6to4サイトからグローバルIPv6インターネット

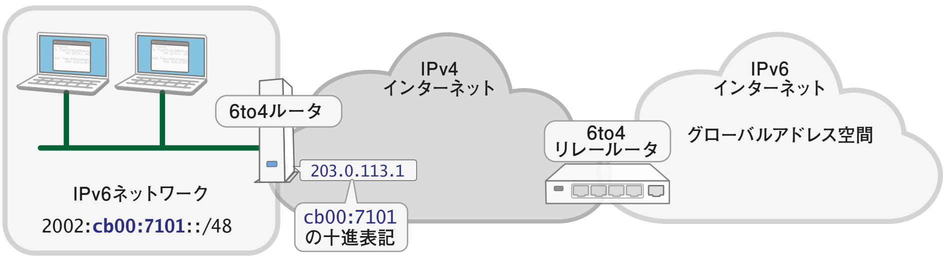
に接続する通信」の両方について考えなければなりません。以降では、RFC 3056での説明にならって、まず6to4サイト同士の接続について考えます。

図22.4は、IPv4インターネットを経由して互いに独立した6to4サイト同士が通信する状況を示したものです。それぞれの6to4サイトの内部では、6to4用のIPv6アドレスが利用されます。このIPv6アドレスは、それぞれの6to4ルータがインターネット接続に使うグローバルIPv4アドレスに基づいたものになっているのがポイントです。



▶ 図22.4　6to4サイト同士の接続

次に、6to4サイトとグローバルIPv6インターネットをつなげる場合です（図22.5）。この場合には、6to4アドレス空間とグローバルIPv6空間を仲介するために、6to4リレールータと呼ばれる機器が必要になります。



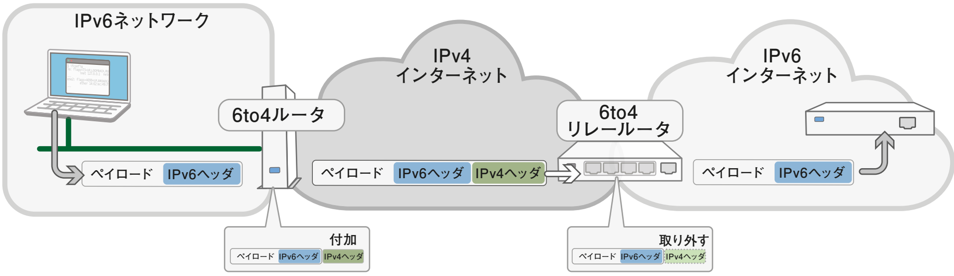
▶ 図22.5　6to4リレールータの利用

NOTE

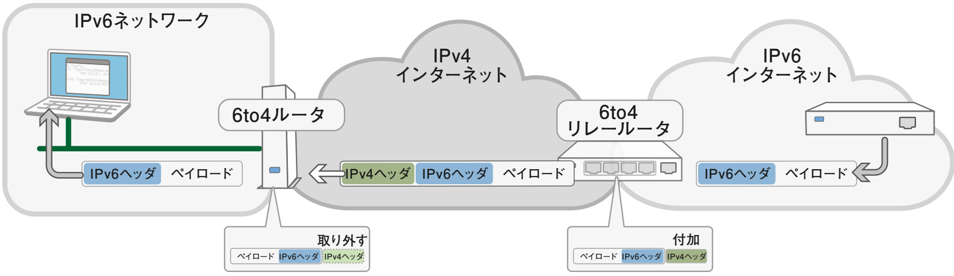
6to4の説明では、この6to4リレールータの存在を前提としたものが少なくありませんが、6to4そのものは6to4リレールータがなくても利用可能である点に注意してください。6to4リレールータは、6to4サイトからのIPv6パケットをグローバルIPv6インターネットと通信できるように仲介するものです。

6to4リレールータを利用したときのカプセル化の流れを図22.6と図22.7に示します。いずれも、6to4サイトの内部にいるユーザがグローバルIPv6インターネットに接続されたサーバと通信を行う場合を示したものです。図22.6が6to4サイト内のユーザからサーバへのパケットの流れ、図22.7がサーバから6to4サイト内へのパケットの流れです。

図22.6では、ユーザから送信されたIPv6パケットを6to4ルータがIPv4でカプセル化して



▶ 図 22.6　ユーザ→6to4 ルータ→6to4 リレールータ→IPv6 ノード



▶ 図 22.7　IPv6 ノード→6to4 リレールータ→6to4 ルータ→ユーザ

います。6to4 ルータによって IPv4 カプセル化された IPv4 パケットは、6to4 リレールータへと転送されます。6to4 ルータからの IPv4 パケットを受け取った 6to4 リレールータは、IPv4 パケットから IPv6 パケットを取り出し、IPv6 パケットとして IPv6 インターネットへと転送します。6to4 リレールータが IPv4 パケットを取り除いて得られた IPv6 パケットは、通常の IPv6 パケットと同様に IPv6 インターネットを転送され、サーバへと到達します。

図 22.7 は、IPv6 で運用されているサーバから 6to4 サイト内のユーザへの通信になります。サーバから送信された IPv6 パケットは、6to4 リレールータへと転送されます。6to4 リレールータは、サーバからの IPv6 パケットに指定された宛先 IPv6 アドレスから、6to4 ルータのグローバル IPv4 アドレスを知ります。それをもとに、サーバからの IPv6 パケットを IPv4 パケットでカプセル化するための宛先 IPv4 アドレスを生成します。カプセル化された IPv4 パケットは、IPv4 インターネットを転送されて、6to4 ルータへと到達します。6to4 ルータは、IPv4 パケットから IPv6 パケットを取り出し、6to4 サイト内にいるユーザへと転送します。

22.1.3 6to4 とエニーキャスト

6to4 で IPv6 インターネットに接続する場合には、6to4 リレールータの IPv4 アドレスが必要です。ユーザ側の 6to4 ルータは、6to4 リレールータの IPv4 アドレスをどのようにして知ればいいでしょうか。

実は、6to4 リレールータへの接続には IPv4 のエニーキャストを利用できます。6to4 で利用

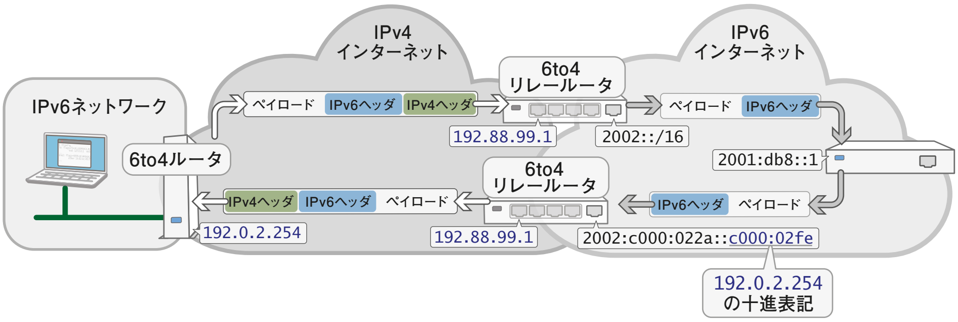
(cc) BY NC SA

するエニーキャストのIPv4アドレスは、RFC 3068[†6]において、`192.88.99.0/24`と規定されています。

NOTE

ユーザが特別な設定を行わなくてもエニーキャストで自動的にトンネルが作成されるという手軽さは、6to4の普及に貢献したと考えられます。実際、6to4の実運用ではエニーキャストが利用される場合が多くありました。ただし、6to4そのものを規定するRFC 3056は、エニーキャストに依存する技術ではないことに注意してください。

6to4でエニーキャストを利用する場合の例を図22.8に示します。図22.8の例では、`192.88.99.1`というIPv4エニーキャストアドレスを利用しています。この同じIPv4アドレスを持つ複数の6to4リレールータが運用されており、6to4ルータから送信されるパケットは最寄りの6to4リレールータへと自動的に転送されるわけです。



▶ 図22.8　エニーキャストを利用する6to4

図22.8の例では、IPv6インターネットでも`2002::/16`のエニーキャストによってサーバから6to4リレールータまで自動的にIPv6パケットが転送されています。

22.1.4　6to4の問題点

6to4は非常に単純なプロトコルであり、ユーザによる設定が少なくて済むという利点がありますが、欠点も多くありました。そのため、2011年には、6to4に関する標準のステータスを「Historical」（すなわち利用を推奨しない）へと変更するという議論がIETFで交わされました。最終的にHistoricalになることはありませんでしたが、その代わりに、6to4の運用に関する注意点を示したRFC 6343[†7]が作成されました。

その後、2015年には、6to4のためのエニーキャストを廃止するという内容のRFC 7526[†8]が発行されています。

6to4の主な欠点としては、以下のようなものが挙げられます。

†6　RFC 3068：C. Huitema, "An Anycast Prefix for 6to4 Relay Routers", 2001年6月
†7　RFC 6343：B. Carpenter, "Advisory Guidelines for 6to4 Deployment", 2011年8月
†8　RFC 7526：O. Troan, B. Carpenter, "Deprecating the Anycast Prefix for 6to4 Relay Routers", 2015年5月

- プライベートIPv4アドレスを扱えず、NATとの相性が悪い
- エニーキャストで運用されることが多く、どのような経路を通過するかわからない
- ユーザが気がつかずに障害に巻き込まれる可能性がある
- セキュリティ上の懸念
- CDNとの相性が悪い

こうした欠点により、IPv6を利用することでIPv4の場合よりも著しく通信品質が悪くなるような状況が、6to4によって引き起こされる可能性もありました。それぞれの項目について下記に要約します。

- **NATとの相性が悪い**

  6to4は、IPv6アドレス中に直接IPv4グローバルアドレスを代入する仕様です。そのため、NATに対応していません。後に提案された、Teredo、ISATAP、6rdなどのトンネリングプロトコルはNAT環境下でも利用できるような仕様になっています。

- **どのような経路を通過するかわからない**

  6to4はエニーキャストを利用しているため、ユーザのパケットがどのような経路を通過して通信を行うのかを制御しにくいという問題があります。たとえば、同じ国内に存在しているサーバとの通信を行うために地球を半周してから戻ってくることもあります。さらに、IPv6で接続されたエンドユーザからは、途中経路上で6to4による通信が行われていてもIPv4においてパケットが通過している経路を知る手段がないことも多く、どのような経路で通信が行われているのかを知ることさえできない場合があります。

- **障害に巻き込まれてもユーザが気づけない**

  6to4リレールータの仕組みを利用していて、ユーザにグローバルアドレスが配られている場合、ユーザは途中経路で6to4が利用されていることに気がつけません。さらに、エニーキャストを利用して設定を行わずに実現が可能であるため、ユーザがまったく気がつかずに6to4ルータを運用してしまっている場合もあります。「IPv6が利用できないはずなのになぜか利用できていて、IPv6の通信が非常に遅い」という状況に遭遇したら、6to4ルータが途中経路に存在していることを疑ってみてもいいかもしれません。

- **セキュリティ上の懸念**

  6to4におけるセキュリティ上の懸念として、6to4を踏み台にしたDDoS攻撃、6to4ネットワークへのDDoS攻撃、6to4を利用した通信の秘匿（spoofing traffic）、IPv4ネットワークから6to4を経由して近隣探索メッセージを送るといった手法がRFC 3964[†9]でまとめられています。

- **CDNとの相性が悪い**

  どのような経路を通過するかわからないという問題と同根ですが、パケットが通過する経路が冗長になる傾向があるため、CDNとの相性が悪くなりがちです。CDNとの相性が

†9 RFC 3964：P. Savola, C. Patel, “Security Considerations for 6to4”, 2004年12月

CC BY NC SA

悪くなると、大手Webサイトの表示が遅くなったり、ソフトウェア自動アップデートやウィルス対策ソフトのパターンファイルのダウンロードが遅くなる場合があります。

なお、IETFにおける6to4に関連するさまざまな検討内容については、すでに触れたRFC 3056、RFC 3068、RFC 3964、RFC 6343、RFC 7526のほか、RFC 2893[†10]、RFC 3904[†11]、RFC 5158[†12]、RFC 6732[†13]の各RFCも参照してください。

22.2　Teredo

Teredoは、6to4の欠点を補うことが可能な仕組みとして、RFC 4380[†14]で規定されたプロトコルです。Microsoft主導で標準化されたことから、Windowsの各種バージョンに標準で搭載されています。

6to4がIPv4パケット上に直接IPv6を乗せるトンネリングプロトコルとして実現されているのに対し、TeredoはUDP上にIPv6パケットを乗せるトンネルとして実現します。これにより、6to4では難しいNAT越えを可能としているのが特長です。

Teredoは、6to4が利用できないNAT環境下でも機能しますが、IPv6インターネットへのネイティブな接続ができないときに使うためのものです。グローバルIPv4アドレスが使える環境であれば、Teredoではなく6to4を使うことを検討してもよいと説明されることもかつてはありましたが、現在は6to4が推奨されなくなっています（22.1節参照）。RFC 4380では、Teredoのことを、あくまで他の方法がないときの最終手段（“IPv6 access of last resort”）として設計されたものと位置づけています。なお、Teredoはトンネルを生成するプロトコルなので、他のトンネル生成プロトコル同様に、DDoSの踏み台になるなどセキュリティ上の懸念があることには注意が必要です。

22.2.1　RFC 3489におけるNATの分類

Teredoは、いわゆる「NAT越え」のためのツールのひとつである旧STUN（RFC 3489[†15]を簡易にしたものだと考えることができます。旧STUNでは、「NAT機器の裏側に設置された機器」と相互にやり取りするための情報を収集するために、NATを次の4種類に分類しています（詳しくは20.2節参照）。

- Full Cone NAT

  NATの外部から特定のIPv4アドレスとポート宛に送信されたパケットは、送信元IPv4アドレスによらずNATの内側へと転送される

---

†10　RFC 2893：R. Gilligan, E. Nordmark, “Transition Mechanisms for IPv6 Hosts and Routers”, 2000年8月

†11　RFC 3904：C. Huitema, R. Austein, S. Satapati, R. van der Pol, “Evaluation of IPv6 Transition Mechanisms for Unmanaged Networks”, 2004年9月

†12　RFC 5158：G. Huston, “6to4 Reverse DNS Delegation Specification”, 2008年3月

†13　RFC 6732：V. Kuarsingh, Y. Lee, O. Vautrin, “6to4 Provider Managed Tunnels”, 2012年9月

†14　RFC 4380：C. Huitema, “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)”, 2006年2月

†15　RFC 3489：J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, 2003年3月

- Restricted Cone NAT
  NATの外部から送信されたパケットは、過去に内部から送信されたIPv4アドレスが送信元であれば、NATの内側へと転送される
- Port-restricted Cone NAT
  NATの外部から送信されたパケットは、過去に内部から送信されたIPv4アドレスとポートが送信元であれば、NATの内側へと転送される
- Symmetric NAT
  NATの内側と外側とで、パケットのアドレスとポートの情報の対応関係をNAT機器で保持する

Teredoは、上記NATの4分類のうち、Symmetric NAT以外の3つで利用可能です。RFC 4380には、NAT機器の設定によりSymmetric NATでもTeredoが利用可能であると書かれていますが、基本的に「できない」と考えてよいでしょう。

RFC 3489は本書が出版された時点では推奨されておらず、上記のNATの4分類についても問題が指摘されています。とはいえ、Teredoの仕組みを理解するには上記のNATの分類について意識しなければならない部分があるので、ここでは上記の分類に基づいて解説を進めます。Teredoそのもののアップデートについては22.2.6項で説明します。

22.2.2 Teredoクライアントによる初期構成

Teredoでトンネル化に関与する機器は、Teredoサーバ、Teredoクライアント、Teredoリレーと呼ばれます。一般のエンドユーザが利用するのはTeredoクライアントです。Teredoクライアントは、Teredoリレーを経由して、IPv6インターネット上のホストと通信します。

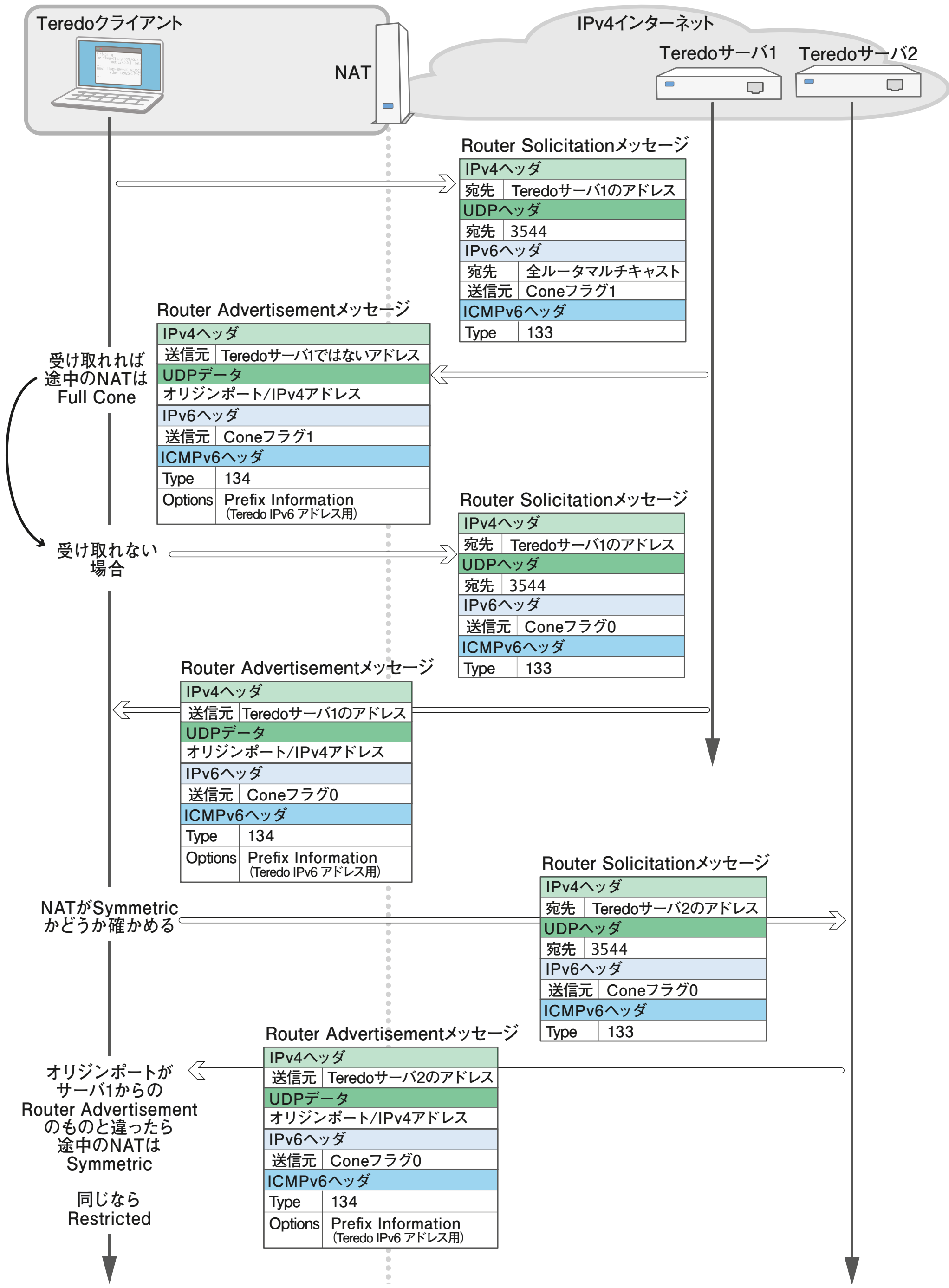
TeredoクライアントがIPv6インターネット上のホストと通信する際の流れは、TeredoクライアントがFull Cone NATの内側にいるのか、それともRestricted Cone NAT（およびPort-Restricted Cone NAT）の内側にいるのかで異なります。そこでTeredoクライアントは、まず「自身が上記の4種類のNATのいずれの内側にいるのか」を事前に判断する必要があります。この判断は、Teredoサーバとの間で、Qualification Procedureと呼ばれる初期構成を実施することで行います。図22.9にQualification Procedureの流れを示します。

Teredoクライアントによる初期構成は、Teredoサーバとのやり取りで開始します。そのため、Teredoクライアントは、あらかじめTeredoサーバのIPv4アドレスを知っている必要があります。Teredoサービスの利用に認証が必要な場合には、適切な認証情報も必要です。

Teredoサーバでは、一般にUDPポート3544でパケットを受信します。Teredoクライアントは、Teredoサーバに向けて、IPv4 UDPにカプセル化されたICMPv6のRouter Solicitationメッセージを送信します。このときのIPv6パケットの送信元アドレスは、Coneフラグと呼ばれるビットを1に設定したリンクローカルユニキャストアドレスです。

Teredoクライアントから、IPv4 UDPのトンネル経由でRouter Solicitationメッセージを受け取ったTeredoサーバは、やはりIPv4 UDPにカプセル化して、Router AdvertisementメッセージをTeredoクライアントに返信します。その際、IPv4パケットの送信元アドレスとしては、TeredoクライアントからのUDPパケットを受信したIPv4アドレスとは別のアドレスを指

(CC BY-NC-SA)



▶ 図22.9　Teredoクライアントによる初期構成

定します。また、Router Advertisementメッセージにおける送信元IPv6アドレスでは、Coneフラグを1に設定します。

ここで、もしTeredoクライアントがFull Cone NATの内側に配置されていれば、このRouter Advertisementメッセージを受け取れます。なぜなら、Full Cone NATでは、外部に送

CC BY NC SA

信したUDPパケットの宛先IPv4アドレスとは別のIPv4アドレスからのUDPパケットもNATの裏側に届くからです。つまり、TeredoサーバからConeフラグに1が設定されたRouter Advertisementメッセージを受け取ったTeredoクライアントは、自身がFull Cone NATの内側に設置されていると判断できます。

Teredoクライアントが利用しているNAT機器がFull Cone NATではない場合には、送信元IPv4アドレスが変更されたTeredoサーバからのRouter AdvertisementメッセージがTeredoクライアントに届きません。Teredoクライアントは、タイムアウト後、今度はConeフラグを0に設定して、Router Solicitationメッセージをカプセル化したIPv4 UDPパケットをTeredoサーバ宛に送信します。このRouter Solicitationメッセージを受け取ったTeredoサーバは、Coneフラグを0に設定してRouter Advertisementメッセージを返します。その際には、Teredoクライアントから受け取ったIPv4 UDPパケットの送信元IPv4アドレスとUDPポートに送信します。

TeredoサーバからConeフラグ0のRouter Advertisementメッセージを受け取ったTeredoクライアントは、自身がSymmetric NATの内側に配置されていないことを確認するために、別のTeredoサーバに対してConeフラグを0にした別のRouter Solicitationメッセージを送信します。Coneフラグが0のRouter Solicitationメッセージを受け取ったTeredoサーバは、Coneフラグが0のRouter AdvertisementメッセージをTeredoクライアントに返信します。もし、別のTeredoサーバから送られてきたメッセージに含まれる送信元ポート番号の情報と、最初のTeredoサーバから送られてきたメッセージの送信元ポート番号が異なる場合には、途中のNATでアドレスとポート番号の対応関係が管理されているということなので、Teredoクライアントが配置されているのはSymmetric NATの内側ということになり、Teredoを利用できないと判断します。

NOTE

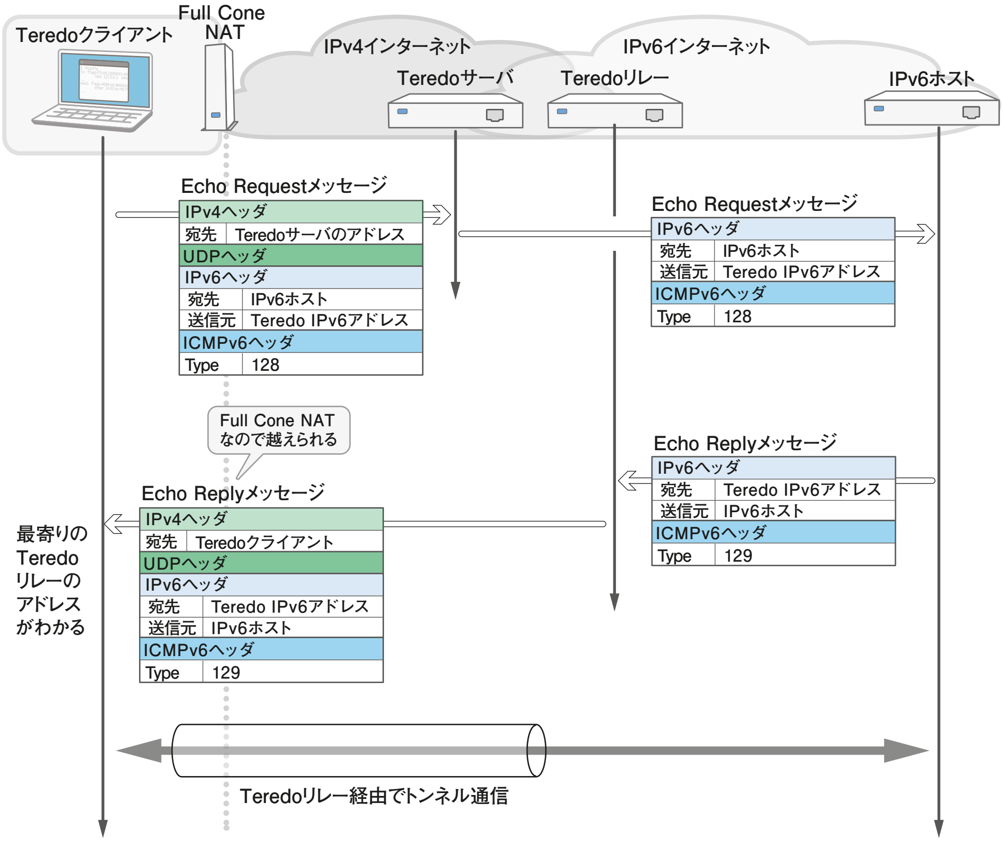
Teredoの問題点として、TeredoクライアントとTeredoサーバとの初期設定により外部とつながるまでに時間がかかる点が挙げられます。TeredoクライアントがPCであれば、そのPCが再起動などを行うたびにTeredoサーバとのやり取りが必要になるでしょう。

22.2.3 TeredoクライアントからIPv6インターネットとの通信（Full Cone NAT）

初期構成によりNAT環境の種類が検出できたら、TeredoクライアントにTeredo IPv6アドレス（22.2.5項参照）が設定されます。このTeredo IPv6アドレスを利用することで、TeredoクライアントはTeredoリレーを経由してIPv6インターネットと通信できるようになります。

Teredoリレーを経由した通信では、通信ごとに適切なTeredoリレーを選ぶ必要があります。Teredoクライアントは、適切なTeredoリレーを知るために、まずICMPv6 Echo RequestメッセージをIPv4 UDPにカプセル化してTeredoサーバへと送信します。IPv4 UDPにカプセル化されたICMPv6 Echo Requestメッセージを受け取ったTeredoサーバは、IPv4 UDPパケットからICMPv6パケットを取り出したうえで、IPv6を利用してIPv6ホストへとICMPv6 Echo Requestメッセージを転送します。このときのICMPv6 Echo RequestメッセージのIPv6送信

cc BY NC SA



▶ 図22.10　Teredoクライアントから IPv6インターネットとの通信（Full Cone NAT）

元アドレスは、Teredoクライアントの Teredo IPv6アドレスです。

ICMPv6 Echo Requestメッセージを受け取ったIPv6ホストは、ICMPv6 Echo Replyを返します。ICMPv6 Echo ReplyメッセージはTeredo IPv6アドレスに向けて送信されますが、その経路はIPv6インターネットに接続された最寄りのTeredoリレーとなります。

Teredoリレーは、IPv6 Teredoアドレスの情報をもとにIPv4 UDPでICMPv6 Echo Replyパケットをカプセル化し、Teredoクライアントへと IPv4 UDPパケットを送信します。Teredo IPv6アドレスから、Teredoクライアントが接続しているNATがFull Cone NATであることがわかるので、Teredoクライアントへ送信される IPv4 UDPパケットの送信元がTeredoリレーのIPv4アドレスのまま送信されてもUDPパケットがNATを越えることができます。

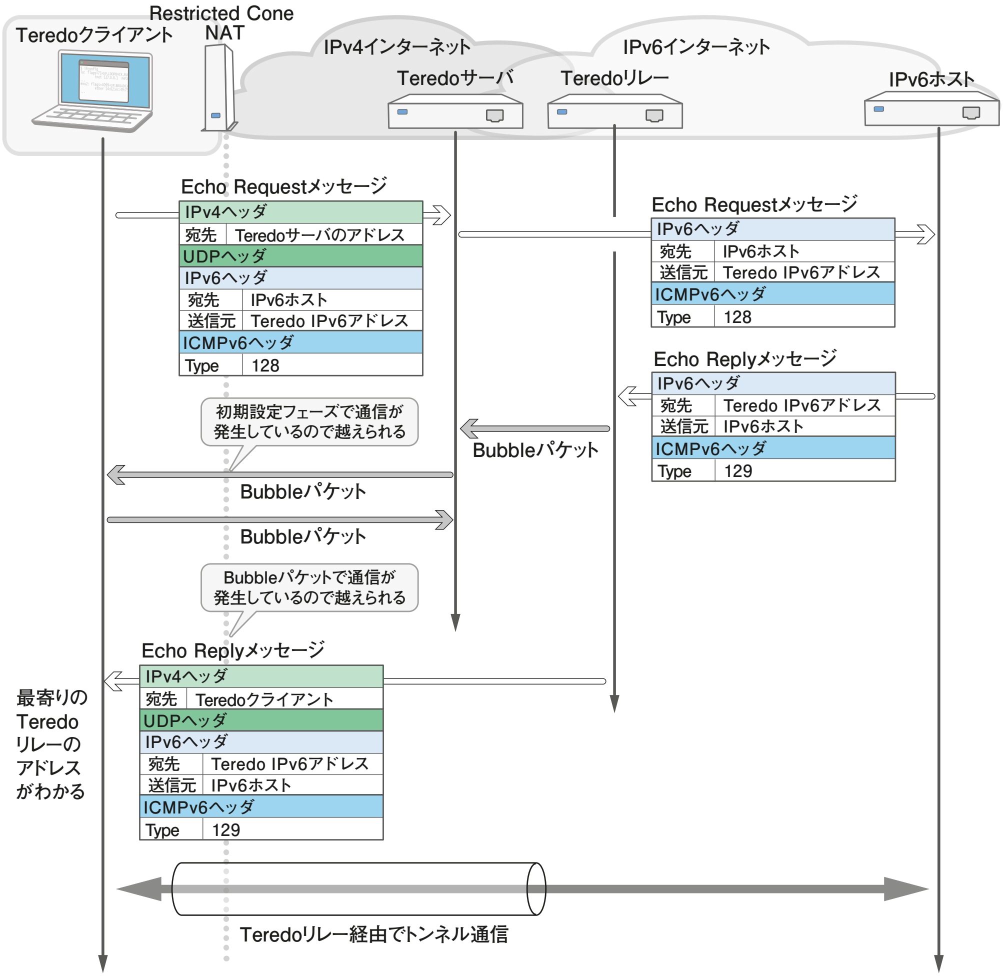
Teredoクライアントは、TeredoリレーからのIPv4 UDPパケットによって、対象とするIPv6ホストに近いTeredoリレーを知ることができます。そのIPv6ホストとの通信を行う際には、そのTeredoリレーへとIPv4 UDPパケットを送るようになります。

Teredoリレーへでは、受け取ったIPv4 UDPパケットからIPv6パケットを取り出して、IPv6ホストへと送信します。

NOTE

Teredoクライアント同士が通信する場合は、これらとは異なる手順で通信が確立されます。詳しくはMicrosoft TechNet[†16]などを参照してください。

cc BY NC SA



▶ 図 22.11　Teredo クライアントから IPv6 インターネットとの通信（Restricted Cone NAT）

22.2.4 Teredo クライアントから IPv6 インターネットとの通信（Restricted Cone NAT）

Teredo クライアントが Restricted Cone NAT の内側に配置されている場合は、同じ送信元からでなければ UDP パケットが NAT 内部へと転送されないので、Full Cone NAT の場合よりも多少複雑です。図 22.11 に、Restricted Cone NAT の場合の Teredo リレーの流れを示します。ポイントは、Teredo サーバの IPv4 アドレスを IPv6 アドレスに埋め込むところです。

Teredo クライアントが IPv4 UDP でカプセル化した ICMPv6 Router Solicitation メッセージを Teredo サーバ向けに送信し、Teredo サーバから IPv6 で ICMPv6 Echo Request が送信され、IPv6 ホストが ICMPv6 Echo Reply を送信するところまでは、Full Cone NAT の場合とまったく同じです。

Full Cone NAT では、送信元 IPv4 アドレスが異なっていても NAT の内側に UDP パケットが

†16　Microsoft TechNet - About Teredo ： https://technet.microsoft.com/ja-jp/library/aa965905.aspx

転送されたので、このICMPv6 Echo Replyメッセージをそのまま Teredoクライアントに送信できました。これに対し、Restricted Cone NATでは、NAT変換テーブルに記載された送信元IPv4アドレスが一致しないUDPパケットが転送されません。そのため、何もせずにTeredoリレーからTeredoクライアントに向けてIPv4 UDPパケットでカプセル化したICMPv6 Echo Replyメッセージを送信してしまうと、途中のNAT機器でそのIPv4 UDPパケットが破棄されてしまいます。

そこでTeredoリレーでは、ICMPv6 Echo Replyをすぐに返信せず、いったんバッファに入れておきます。そして、NAT機器に適切なNAT変換テーブルを作成するために、Teredoサーバ経由でTeredoクライアントからTeredoリレーに向けたUDPによる通信を発生させます。この通信は、Bubbleパケットと呼ばれます。

Bubbleパケットは、まずTeredoリレーからTeredoサーバ宛に送信されます。Teredoリレーが適切なTeredoサーバに向けてBubbleパケットを送信できるのは、Teredo IPv6アドレスにTeredoサーバのIPv4アドレスが埋め込まれているからです。

TeredoリレーからのBubbleパケットを受信したTeredoサーバは、Teredoクライアントに向けてBubbleパケットを転送します。TeredoサーバがTeredoクライアントへと送信するBubbleパケットには、TeredoリレーのIPv4アドレスとUDPポート番号が含まれています。このBubbleパケットがNATを越えられるのは、初期構成フェーズで、TeredoクライアントからTeredoサーバに向けた通信が発生しているからです。

Teredoリレーが送信したBubbleパケットをTeredoサーバ経由で受け取ったTeredoクライアントは、Teredoリレーに向けてIPv4 UDPでBubbleパケットを送信します。このBubbleパケットによって、NAT機器内に、TeredoクライアントとTeredoリレーを結ぶNAT変換テーブルエントリが作成されます。

TeredoクライアントからのIPv4 UDPによるBubbleパケットを受け取ったTeredoリレーは、バッファ内で待機させていたICMPv6 Echo ReplyメッセージをIPv4 UDPにカプセル化してTeredoクライアントへと送信します。こうして、TeredoクライアントからIPv4 UDPにカプセル化されたIPv6パケットをTeredoリレー経由で目的のIPv6アドレス宛に送信できるようになります。

22.2.5 Teredo IPv6アドレス

Teredo IPv6アドレスの構成を図22.12に示します。



▶図22.12　Teredo IPv6アドレス

ポートフィールドには、Teredoクライアント側でのUDPポート番号のビットを反転させて難読化されたものが格納されます。クライアント側のIPv4アドレスも、IPv4アドレスのビッ

トを反転させて難読化されたものが格納されます。

フラグフィールドは図22.13のようなフォーマットとなっています。

	64						70	71	72 … 79	
...	C	0	0	0	0	0	U	G	0	...

▶ 図22.13 Teredo IPv6アドレスのフラグフィールド

先頭のCがConeフラグです。TeredoクライアントがFull Cone NATの内側であると認識された場合に利用されるIPv6アドレスではCが1となります。Full Cone NATでなければ0です。

UGの部分は、IPv6アドレスのModified EUI-64フォーマットに準拠しており、グローバルではないユニキャストであることを示す00となっています。

フラグの他の部分はすべて0です。つまり、RFC 4380の仕様でフラグフィールドが取りうる値は、Coneビットが設定されていない0x0000か、Coneビットが設定されている0x8000の2種類のみです。

22.2.6 Teredoのアップデート

Teredoは、RFC 5991[†17]とRFC 6081[†18]によってアップデートされています。

RFC 5991は、Teredoのセキュリティを向上させるためのアップデートです。RFC 4380のTeredoでは、IPv6アドレスに13ビットの未使用フィールドがありましたが、外部からのアドレススキャンを避けるために、その部分にランダムな値を使うことが推奨されるようになりました。

RFC 5991によるもう1つのアップデートは、Coneビットの廃止です。Teredoでは、NATの種類を推測したうえで、その種類がTeredo用IPv6アドレスに反映されていましたが、その情報が外部の攻撃者にヒントを与えることになるので廃止されました。

RFC 5991によるアップデートでは、Teredoクライアントの初期構成においてConeビットを1にしてRouter Solicitationメッセージを送るフェーズを行わないことを求めています。また、TeredoリレーもConeビットを無視するようにすべきであるとしています。これらにより、Coneビットが1となるTeredo IPv6アドレスは採用されなくなります。

ただし、RFC 5991では、古いTeredoクライアントとの後方互換性などのために、TeredoサーバはConeビットを従来どおりに処理しなければならないとしています。

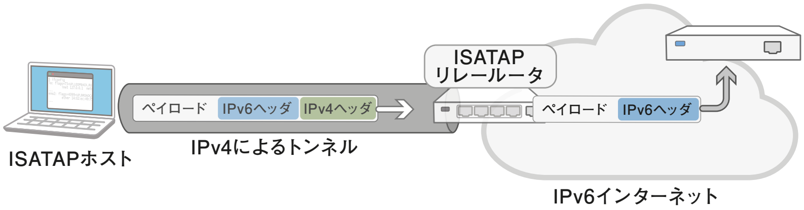
RFC 6081では、RFC 4380で定義されているNAT対応よりもさらに幅広いNAT対応のための拡張が定義されています。

22.3 ISATAP

ISATAPは、自動的にIPv4トンネルを構築してIPv6インターネットへの接続性を提供する

†17 RFC 5991：D. Thaler, S. Krishnan, J. Hoagland, “Teredo Security Updates”, 2010年9月
†18 RFC 6081：D. Thaler, “Teredo Extensions”, 2011年1月

ためのプロトコルです。Informational な RFC として、RFC 5214[†19]で定義されています。
ISATAP は、図 22.14 のような構成で動作します。



▶ 図 22.14　ISATAP

クライアントとなる ISATAP ホストは IPv4 インターネットのみに接続していますが、ISATAP ルータを経由して IPv6 インターネットとの通信も可能になります。ISATAP ルータは、IPv4 と IPv6 両方のインターネットに接続されており、ISATAP クライアントから IPv4 トンネル経由で受け取った IPv6 パケットを IPv6 インターネットへと転送します。IPv6 インターネットから ISATAP ホストに対する IPv6 パケットを受け取った ISATAP ルータは、IPv4 トンネル経由で ISATAP ホストへと届けます。

このときの IPv4 トンネルは、6to4 と同様に、IPv4 のプロトコル番号フィールドに IPv6 を示す 41 番を設定することで構築されます。ただし、動作は 6to4 と異なります。エニーキャストを利用して自動的なトンネルを実現する 6to4 に対し、ISATAP では以下の 3 種類の方法でトンネルを構築する相手を発見します。

- DNS に登録された `isatap.example.com` のような名前
- IPv4 アドレスの手動設定
- DHCPv4 のベンダー固有オプション

6to4 のようにプロトコルに依存する IPv6 プレフィックスを利用せず、通常の IPv6 ユニキャストプレフィックスを利用可能なので、ISATAP ホストが利用する経路が明確であるという点も 6to4 との大きな違いです。さらに、6to4 と違って、ISATAP はプライベート IPv4 アドレスにも対応しています。

22.3.1　ISATAP IPv6 アドレス

ISATAP ホストが利用する IPv6 アドレスは、以下の手順で生成されます。

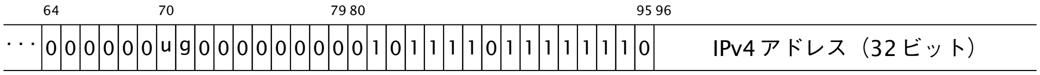
1. ISATAP ホストが ISATAP インターフェース識別子を生成
2. ISATAP ホストが ISATAP インターフェース識別子を利用してリンクローカルユニキャストアドレスを生成
3. ISATAP ホストが ISATAP ルータに Router Solicitation メッセージを送信する
4. ISATAP ルータから ISATAP ホストへ、Router Solicitation メッセージの返答として Router Advertisement メッセージが送信される

[†19] RFC 5214：F. Templin, T. Gleeson, D. Thaler, “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)”, 2008 年 3 月

CC BY NC SA

5. ISATAPホストがRouter Advertisementメッセージの情報を利用してIPv6アドレスを自動生成

ISATAPでは、ISATAPホストで利用されるIPv6アドレスの下位64ビットをISATAPインターフェース識別子としています。ISATAPインターフェース識別子は、以下のようなフォーマットです。ISATAPホストのIPv4アドレスがISATAPインターフェース識別子に埋め込まれているのがポイントです。



▶ 図22.15　ISATAP IPv6アドレスの下位64ビット

これはModified EUI-64形式（7.4.1項参照）のIPv6アドレスです。OUIとしてはIANAの持つ`00-00-5e`が利用され、`0xfe`がそれに続き、その後にIPv4アドレスをつなげる形式になっています。このModified EUI-64形式が利用されるIPv4アドレスがグローバルに一意性があるIPv4アドレスである場合には、図22.15の`u`は1となり、そうでない場合には`u`は0となります。`g`はindividual/groupビットを示します。

ISATAPでは、ISATAPホストのリンクローカルアドレスを生成します。リンクローカルユニキャストアドレスの`fe80::/10`に対して、下位64ビット分のISATAPインターフェース識別子を加えたものが、ISATAPホストのリンクローカルアドレスになります。

ISATAPホストは、IPv4トンネルを経由して、生成されたリンクローカルアドレスによるRouter SolicitationメッセージをISATAPルータに送信します。Router Solicitationメッセージを受け取ったISATAPルータは、IPv4トンネルを経由してISATAPホストにRouter Advertisementメッセージを返信します。

ISATAPホストは、受け取ったRouter Advertisementメッセージに記載されたIPv6プレフィックス情報とISATAPインターフェース識別子を組み合わせてIPv6アドレスを自動生成します。ISATAPルータは、ISATAPホストのIPv4アドレス情報を含むIPv6アドレスから、適切なISATAPホストへとIPv4トンネルを経由してIPv6パケットを送信できます。

22.3.2 セキュリティ上の懸念

2011年に、JPRSからWindows VistaおよびWindows 7のISATAPルータ自動発見方式の脆弱性が公表されています[†20]。Windows VistaとWindows 7では、ISATAPホストとなってISATAPルータを自動発見する機能がデフォルトで有効に設定されており、ユーザが意図しなくても通信時にDNSを利用して`isatap`という名前のホストを検索します。このとき、同実装では、第三レベルドメイン名までドメイン名空間を遡って検索します。たとえば、`subdomain.example.co.jp`というドメイン名に属するISATAPホストがISATAPルータを検索するとき、以下の3つが検索対象となります。

- `isatap.subdomain.example.co.jp`

[†20] 「Windows Vista/7におけるISATAPルーター自動発見方式の脆弱性について」: https://jprs.jp/tech/notice/2011-03-17-isatap-router-auto-discovery.html

- isatap.example.co.jp
- isatap.co.jp

上記3つのうち、最後のisatap.co.jpはexample.co.jpと管理主体が異なる可能性が高く、結果として意図しないISATAPルータを経由して通信が行われてしまう可能性がありました。この対策のため、JPRSではisatapを予約ドメイン名とし、登録できないものとしています。これは、ISATAPそのもののプロトコル上の脆弱性というよりも、プロトコルと実装の組み合わせによる脆弱性です。こういった脆弱性の発生事例も、IPv4/IPv6共存技術を利用するうえでの落とし穴を知るという意味では重要でしょう。

もちろん、6to4などと同様にトンネル化に伴うセキュリティリスクもあるので注意が必要です。たとえば、ISATAPルータが任意のIPv4アドレスからのトンネルを受け付ける設定にしていると、攻撃の踏み台にされる可能性があります。トンネル技術全般のセキュリティリスクに関しては22.7節で改めて説明します。

6over4

6to4のようにエニーキャストを使うのではなく、IPv4マルチキャストを利用したトンネル技術によってIPv6インターネットへの接続性を確保しようという**6over4**という技術があります。6over4は、RFC 2529[†21]で規定されています。6to4など他のいくつかのIPv6用トンネル技術とともに、ISATAPなどの技術が考察されるうえで影響を与えたプロトコルだといえます。

しかし、6over4はまったくと言っていいほど実運用で使われていません。その理由は、6over4がIPv4マルチキャストに依存することにあります。グローバルインターネットでのIPv4マルチキャストそのものが利用されていないことから、6over4を実際に利用するのも困難というわけです。

22.4　6rd

6rdは、6to4同様に、ISPから提供されているIPv4ネットワークを利用してユーザがIPv6インターネットへ接続できるようにするプロトコルです。6rdは、IPv4ネットワークを利用してIPv6インターネットへと接続するステートレスなトンネリングプロトコルであり、本書執筆時点では実用性が有望視されている技術です。

6rdは、2007年に、フランスのISPであるFreeにて6to4を改造することで開発されました。2010年の1月にInformationalとして発行されたRFC 5569[†22]では、5週間という短い期間で、Freeが150万契約の顧客に対してIPv6接続可能な環境を提供したことが紹介されています。6rdの詳細なプロトコル仕様を規定するStandard TrackのRFCは、2010年8月に発行さ

[†21] RFC 2529：B. Carpenter, C. Jung, “Transmission of IPv6 over IPv4 Domains without Explicit Tunnels”, 1999年3月

[†22] RFC 5569：R. Despres, “IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)”, 2010年1月

CC BY NC SA

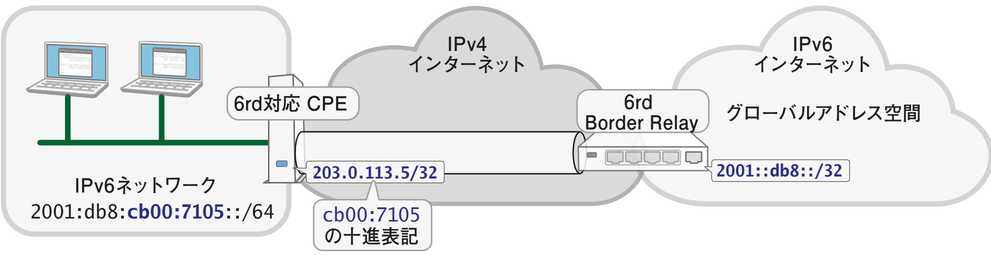
れたRFC 5969[†23]です。

6rdはISPが主体となってIPv6インターネットへの接続性を提供するモデルです。これは、6to4がISPの助けを借りずにユーザが独自にIPv6ネットワークへと接続可能なモデルだったのとは対照的だといえるでしょう。ISPのIPv6アドレスプレフィックスを利用して運用できるので、あらかじめ決められたIPv6プレフィックスを利用する6to4のようにどのような経路をたどるかわからなくなったり、そもそも到達性を確保できるかどうかもわからない状況になったりせず、経路の制御がしやすいという特徴があります。

ルータにおける負荷が少なく、CGN（Carrier Grade NAT）などの環境下でも利用可能であり、全体的に6to4が抱える各種欠点を改善した設計になっている6rdは、各種通信機器ベンダーによって実装され、キャリアでも採用されるようになりました。日本でも、2010年4月よりYahoo! BBで6rdサービスが提供開始されています[†24]。

22.4.1 6rd概要

6rdは、ISPが用意したIPv6プレフィックスと、家庭に設置されるSOHOルータのようなCPE（Customer Premises Equipment）にISPが割り当てたIPv4アドレスを利用して、自動的にトンネルを生成するプロトコルです。CPEと6rd Border Relayの間で通信ができればよいので、CPEに割り当てられるIPv4アドレスが`10.0.0.0/8`などのプライベートIPv4アドレスでも利用可能です。図22.16に6rdの概要を示します。



▶ 図22.16　6rd概要

CPEは、6rd Border RelayのIPv4アドレスや、後述する6rd delegated prefixなどの情報を、DHCPもしくはPPPoEなどから自動的に取得します。家庭内IPv6ネットワークから送信されたIPv6パケットは、ISPによるIPv4ネットワークをトンネルによって通過し、6rd Border Relayで脱カプセル化されてIPv6インターネットへと送信されます。

IPv6インターネットから家庭内IPv6ネットワークへは、6rd Border Relayを経由してIPv4へとカプセル化され、ISPによるIPv4ネットワークを通過し、CPEでIPv4パケットから脱カプセル化されてIPv6パケットが届きます。

---

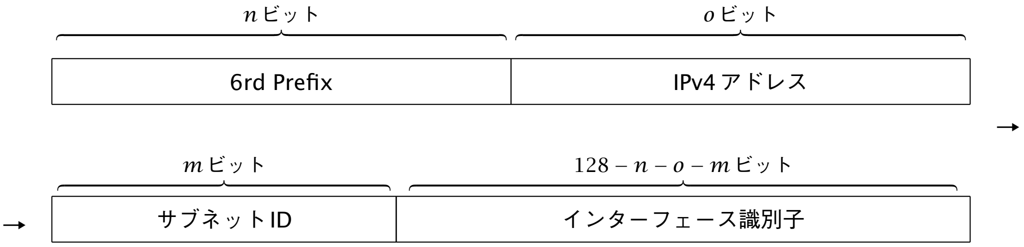
†23 RFC 5969：W. Townsley, O. Troan, “IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) – Protocol Specification”, 2010年8月

†24 「ソフトバンクグループによるIPv6インターネットサービスの提供について」：`https://www.softbank.jp/corp/group/sbb/news/press/2010/20100223_01/`

CC BY NC SA

22.4.2　6rdのIPv6アドレスフォーマット

6rdで利用されるIPv6アドレスの構成を図22.17に示します。6rd prefixとCPEに付けられているIPv4アドレスは6rd delegated prefixと呼ばれます。この6rd delegated prefixをもとに、各パケットが適切なCPEまで転送されます。



▶図22.17　6rdのIPv6アドレスフォーマット

6rdのIPv6アドレスにおけるサブネットID部分やインターフェース識別子部分は3.7節で紹介したIPv6グローバルユニキャストアドレスのサブネットIDと同じです。このため、6rdのIPv6アドレスを外部から見ると、普通のIPv6グローバルユニキャストアドレスと見分けがつきません。

22.5　4rd

4rdは、IPv4パケットをIPv6でカプセル化するステートレスなプロトコルとして提案されました。IPv4インターネットへの接続性しかない環境でIPv6インターネットへの接続性を提供する6rdとは逆の動作をするプロトコルだといえます。

6rdは、IPv4インターネットが主流のインターネットにおいてIPv6インターネットへの接続性を確保するためのプロトコルであり、IPv6が普及する前の環境での利用が想定されています。これに対し、4rdは、IPv6が普及したあとの環境を想定したプロトコルです。4rdを利用することで、たとえばバックボーンネットワークをIPv6のみで構築することが可能になります。4rdの必要性については、“Motivations for Carrier-side Stateless IPv4 over IPv6 Migration”と題されたInternet Draft[†25]が参考になるでしょう。

なお、4rdは6rdの逆の発想であると同時に、CGN運用形態のひとつとして、後述するA+Pの思想も含んでいます。24.4節で詳しく説明します。

6rdと4rdの「rd」の意味

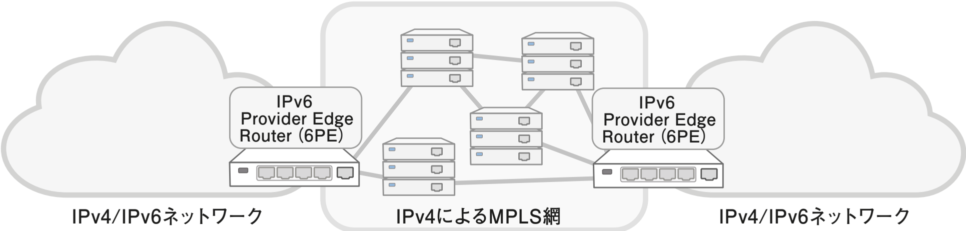
公式には、6rdのrdはrapid deployment、4rdのrdはresidual deploymentと、それぞれ異なる単語を表しています。とはいえ、4rdという名称は6rdを意識して付けられたものであることは間違いないでしょう。

[†25] M.Boucadair, Ed., S.Matsushima, Y.Lee, O.Bonness, I.Borges, G.Chen, “Motivations for Stateless IPv4 over IPv6 Migration Solutions”, 2012年11月（2013年3月に期限切れ）, https://tools.ietf.org/html/draft-ietf-softwire-stateless-4v6-motivation-05

ちなみに、6rdの「rd」というのは、6rdの提案者であるRemi Despres氏の名前のイニシャルに由来しているのではないかという推測もあります（Remi Despres氏は、4rdのドラフトの著者の一人でもあります）。

22.6 6PE

6PE（IPv6 Provider Edge router）は、既存のIPv4 MPLS網を変更せずにそのまま利用し、エッジルータで対応することによってIPv6ネットワーク同士を接続する技術です。RFC 4798[†26]で規定されています。図22.18に6PEの概要を図示します。



▶ 図22.18　6PE概要

6PEルータは、IPv4によるMP-BGPを利用してIPv6の経路情報を交換します。BGPのNext Hopフィールドを利用して6PEルータのIPv4アドレスをMP-BGPでのIPv6経路交換に使うことで、トンネルなどの設定を行わずに、IPv4 MPLS網でLSP（Label Switched Path）が構築できるという仕組みです。

22.7 トンネル技術とセキュリティ

トンネル技術には、どの手法を採用する場合であっても共通して考慮が必要になるようなセキュリティ上の課題がいくつかあります。14.10節でも説明しましたが、改めて要点をまとめると次のようになります。

- ファイアウォール製品などが未対応のトンネル技術がセキュリティ機能の迂回に利用される可能性がある
- 送信元を偽装する目的でトンネル技術が利用される可能性がある
- 外部から内部への攻撃にトンネル技術が利用される可能性がある
- トンネルで利用する特別なアドレスから利用環境をプロファイリングされる可能性がある
- トンネルサーバの設定を変更することによるMan-in-the-Middle攻撃の可能性がある

†26 RFC 4798：J. De Clercq, D. Ooms, S. Prevost, F. Le Faucheur, “Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)”, 2007年2月

[書籍版では空白のページです]

cc BY NC SA

第23章

IPv4/IPv6変換技術

本章で解説するのは、IPv4からIPv6へと変換したり、IPv6からIPv4へと変換したりする変換技術です。IPv6接続性を提供することを目的としたトンネル技術とは異なり、変換技術は各ユーザ環境をデュアルスタックにせずにIPv4とIPv6両方のインターネットを利用可能とします。たとえば、エンドユーザが「IPv4インターネットと通信をしているつもり」でIPv6インターネット上の相手と通信できる仕組みや、逆にIPv6からIPv4へと通信する仕組みがあります。

23.1 IPv4/IPv6変換の枠組み

2021年現在、IPv6の利用が増えてはいるものの、インターネットの大部分はIPv4で構成されており、さまざまなサービスを提供しているサーバの多くはIPv4のみで運用されています。そのため、IPv6のみで構成されているシングルスタックネットワークにおいて、何らかの手段でIPv4インターネットとの通信が要求される場合があります。IPv4とIPv6には直接的な互換性がないので、IPv6シングルスタックネットワークでIPv4インターネットとの通信を行うためには、IPv6による通信をIPv4の通信へと変換する仕組みが必要です。

本書執筆時点では、IPv6とIPv4を変換するシナリオとして、RFC 6144[†1]が提案されています。RFC 6144で定義されているIPv4/IPv6変換フレームワークでは、以下の5つの構成要素が必要であるとしています。

- IPアドレスの変換
- IPパケットおよびICMPパケットの変換
- 変換ステートの保持
- NAT64とDNS64
- アプリケーション用のALG（Application Layer Gateway）

IPv4/IPv6変換では、IPv4アドレスがどのようなIPv6アドレスに変換され、逆にIPv6アドレスがどのようなIPv4アドレスに変換されるのかを定義する必要があります。それらを大別

†1 RFC 6144：F. Baker, X. Li, C. Bao, K. Yin, “Framework for IPv4/IPv6 Translation”, 2011年4月

すると、IPv6 アドレスに IPv4 アドレス情報を埋め込むステートレスな変換（RFC 6052[†2]）による方法と、IPv4 アドレスと IPv6 アドレスの変換をステートフルに行う方法があります。どのような方法を採用するかは、用途や想定される環境によって変わります。

IP パケットおよび ICMP パケットの変換については、SIIT（Stateless IP/ICMP Translation Algorithm）が RFC 7915[†3] として標準化されています。RFC 7915 は RFC 6145[†4] を上書き廃止した RFC であり、ステートレスな変換アルゴリズムを定義していますが、ステートフルな変換に関する考察も含まれています。SIIT に関しては 23.2 節で説明します。

NAT64 は、IPv4/IPv6 変換を実現するための具体的な仕組みであり、RFC 6146[†5] で規定されています。IPv4 プライベートアドレスと IPv4 グローバルアドレスを変換する通常の NAT に対し、IPv6 アドレスと IPv4 アドレスを変換する NAT64 では、特殊なステートの保持が要求されます。NAT64 に関しては、23.3 節で説明します。

NOTE

IPv4 グローバルアドレスと IPv4 プライベートアドレスの変換を行う通常の NAT は、NAT44 と表現されることがあります。

インターネットでの一般的な通信では、名前解決の結果として得られる IP アドレスを持つ相手と通信を行うことになるので、DNS を利用した名前解決においても IPv4/IPv6 変換が必要です。IPv4 のための A レコードを IPv6 の AAAA レコードへと変換する DNS64 は RFC 6147[†6] で定義されています。DNS64 に関しては、23.3.3 節で説明します。

さらに、ペイロード中に IPv4 アドレスに関する情報を含むプロトコルは、IPv4/IPv6 変換だけでは通信できません。そのような場合には ALG（Application Layer Gateway）が利用されます。ALG は NAT44 でも利用されますが、IPv4/IPv6 変換では NAT44 の場合よりも複雑になることがあります。たとえば、FTP では IPv4 と IPv6 とで異なる手法によりパッシブモードを実現しているので、ALG がその差異を吸収する必要があります。IPv4/IPv6 変換のための ALG に関しては、23.5 節で説明します。

23.1.1　廃止された NAT-PT

IPv4/IPv6 変換は、かつては NAT-PT という仕組みで議論されていました。NAT-PT は RFC 2766[†7] で規定されています。しかし、さまざまな問題があったことから、2007 年に廃止されています。さらに、2011 年に発行された RFC 6052、RFC 6144、RFC 6145、RFC 6146、RFC

---

†2　RFC 6052：C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, X. Li, “IPv6 Addressing of IPv4/IPv6 Translators”, 2010 年 10 月

†3　RFC 7915：C. Bao, X. Li, F. Baker, T. Anderson, F. Gont, “IP/ICMP Translation Algorithm”, 2016 年 6 月

†4　RFC 6145：X. Li, C. Bao, F. Baker, “IP/ICMP Translation Algorithm”, 2011 年 4 月

†5　RFC 6146：M. Bagnulo, P. Matthews, I. van Beijnum, “Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers”, 2011 年 4 月

†6　RFC 6147：M. Bagnulo, A. Sullivan, P. Matthews, I. van Beijnum, “DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers”, 2011 年 4 月

†7　RFC 2766：G. Tsirtsis, P. Srisuresh, “Network Address Translation - Protocol Translation (NAT-PT)”, 2000 年 2 月

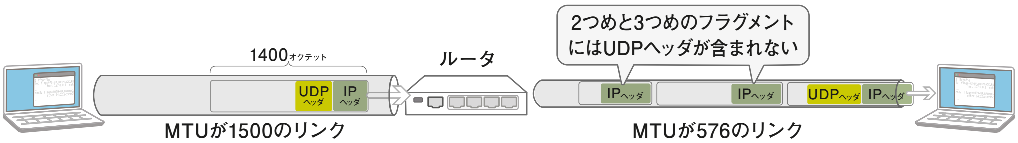
CC BY NC SA

6147、RFC 6384[†8]のRFC群によって、NAT-PTは事実上置き換えられました。これらのプロトコルによって、RFC 2765[†9]で定義されていたSIITも更新されています。

RFC 2765とRFC 2766の新しいRFC群による更新を整理すると下記のようになります。

- RFC 2765のSIITが更新されてRFC 6145に。さらにRFC 6145は2016年に上書き廃止されてRFC 7915に
- IPv4/IPv6変換のフレームワークをRFC 6144で解説
- IPv4/IPv6変換時のIPアドレス変換に関してRFC 6052で解説
- RFC 2766のIPv4/IPv6変換のうち、ステートフルな部分がNAT64としてRFC 6146に
- RFC 2766のDNS ALGに関する部分が、DNS64としてRFC 6147に
- RFC 2766で解説されているFTP ALG対応が、RFC 6384に

NAT-PTが抱えていた問題は、RFC 4966[†10]としてまとめられています。問題の一例としては、フラグメンテーションの扱いが挙げられます。たとえば、TCPやUDPなどのトランスポート層情報を活用するステートレスな変換機が、1番めのフラグメント以降のフラグメントを変換できません。これは、図23.1のようにトランスポート層ヘッダが最初のフラグメントのみに含まれ、それ以降のフラグメントには含まれないことから、各フラグメントを見てもポート番号がわからないことに起因する問題です。フラグメンテーションに対応するためには、何らかのステートを保持し、断片化された複数のパケットを関連づける必要があります。



▶ 図 23.1　IPv4 フラグメンテーションの例

この問題はNAT-PTに限った話ではなく、変換機やトランスポート層におけるポート番号を利用する仕組み全般に適用されます。実際、RFC 3027[†11]では、NATにおけるIPフラグメンテーションの処理における同様の問題が紹介されています。

RFC 4966には、DNS-ALGがNAT-PTの一部である必要があるかといった疑問点も記載されています。これについては、NAT-PTに代わる枠組みではNAT64とDNS64が別々のプロトコルとして定義されています。たとえば、24.6節で紹介する464XLATは、SIITとNAT64のみを要求しており、DNS64なしでも利用可能な設計になっています。

新しい枠組みは、RFC 4966で挙げられている問題点を考慮したうえで規定されているものですが、すべての問題が解決しているわけではありません。たとえば、プロトコル中にIPアド

---

†8 RFC 6384：I. van Beijnum, “An FTP Application Layer Gateway (ALG) for IPv6-to-IPv4 Translation”, 2011年10月

†9 RFC 2765：E. Nordmark, “Stateless IP/ICMP Translation Algorithm (SIIT)”, 2000年2月

†10 RFC 4966：C. Aoun, E. Davies, “Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status”, 2007年7月

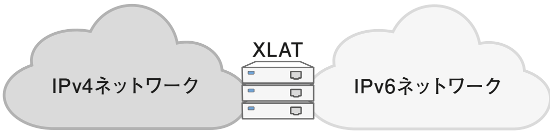
†11 RFC 3027：M. Holdrege, P. Srisuresh, “Protocol Complications with the IP Network Address Translator”, 2001年1月

CC BY NC SA

レス情報が埋め込まれているのでALGでの対応が必要である点は、NAT64でも同様です。また、DNSSECへの対応についても、DNS64はNAT-PTと同様の問題を抱えています†12。

23.2　SIIT

SIITは、図23.2のように、**XLAT**と呼ばれるIPv4/IPv6変換機器がIPv4パケットとIPv6パケットの変換を行う構成になっています。



▶図23.2　IPv4/IPv6変換モデル

IPv4/IPv6変換といって真っ先に思い浮かぶのは、送信元や宛先IPアドレスの変換であったり、TCPとUDPなどのトランスポート層プロトコルのポート番号の変換であったりといった、IPv4におけるNATのような機能かもしれません。しかし、SIITそのものでは、そのような変換については定義していません。SIITでは、IPv4およびIPv6パケットそのものをどのように変換するかや、ICMPをどのように扱うべきかといった内容が規定されています。IPv4とIPv6はまったく異なるプロトコルなので、IPv4/IPv6変換ではIPv4プロトコルとIPv6プロトコルの差異を考慮する必要があるのです。

まず、基本的なところでは、どのIPv4ヘッダフィールドがどのIPv6ヘッダフィールドに対応しているのかが記述されています。変換時に、IPv4とIPv6のプロトコルの違いがあるフィールドもあります。たとえば、IPv4のTotal LengthフィールドはIPv4ヘッダとオプションの長さを含む値ですが、IPv6のPayload Lengthフィールドの値にはIPv6ヘッダの長さは含まれません。

ICMPの扱いも難しいポイントです。そもそも、IPv4ではIGMPとして別のプロトコルであったメッセージがIPv6ではICMPv6に含まれますし、IPv4のICMPとIPv6のICMPv6とでは仕組みが異なるので、そのままでは変換が不可能なメッセージもあります。たとえば、IPv4とIPv6では最小MTUが異なるので、IPv4側からのICMPv4 Packet Too Bigメッセージが示すMTUが、IPv6の最小値である1280を下回る可能性があります。

また、IPv4/IPv6変換機はルータとして動作します。そのため、パケットを変換して転送するときにはHop LimitもしくはTTLを1減らします。Hop LimitもしくはTTLが0になった場合、ICMPv4のTTL Exceed、もしくはICMPv6のHop Limit Exceedの各エラーメッセージが送信されます。

IPv4とIPv6でのフラグメンテーションの違いにも注意が必要です。IPv4ではフラグメンテーションが途中ルータでも起こりますが、IPv6では送信元においてのみフラグメンテーションが行われます。フラグメンテーションの処理とPath MTU Discoveryは、SIITにおいて熟慮が必要な要素だといえます。

†12　RFC 2766のSection 7.5参照。

SIITの制約をまとめると以下のようになります。

- IPv4ヘッダのオプションは変換されない
- フラグメントヘッダを除くすべてのIPv6拡張ヘッダは変換されない
- UDPチェックサムが0となっているフラグメント化されたIPv4 UDPパケットの変換は行われない
- フラグメント化されたICMPv4およびICMPv6パケットは変換されない

23.2.1 RFC 2765による旧SIITとの違い

RFC 7915によって廃止されたRFC 6145は、2000年に発行されたRFC 2765を置き換えるものです。RFC 6145のSIITと、RFC 2765の旧SIITとでは、内容が大幅に変わっています。

RFC 2765には、想定されるシナリオや適用範囲なども書かれていますが、それらはフレームワークを説明するRFC 6144として別になりました。IPアドレスフォーマットの変換に関しても、RFC 6052として別になっています。さらに、RFC 2765ではステートフルな変換の詳細についても記述されていますが、これもNAT64のRFC 6146として別になりました。

その他、フラグメントが関連するパケットやICMPエラーの処理も大きく変わっています。

23.3 NAT64とDNS64

NAT64は、IPv6ヘッダのIPv4ヘッダへの変換、IPv4ヘッダのIPv6ヘッダへの変換、IPv4 ICMPパケットのICMPv6への変換、必要に応じたTCPやUDPヘッダの書き換えを行います。場合によっては、FTPなどの変換（Application Level Gateway）も行います。

DNS64は、IPv4のみで運用されている「名前」を、IPv6で通信できるように変換するための仕組みです。

NAT64とDNS64は、2つの異なる技術ですが、一般的には組み合わせて使われます[†13]。両者の組み合わせは、NAT64+DNS64表記されることがあります。

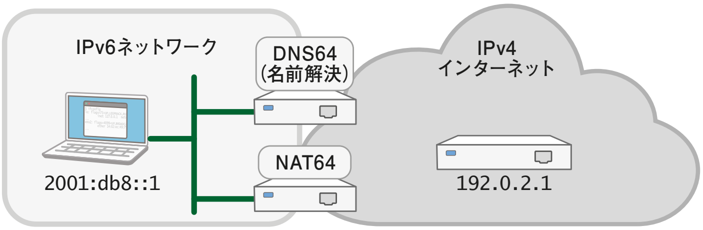
NAT64とDNS64を使うと、エンドユーザ向けのネットワークをIPv6のシングルスタックにしつつ、IPv4インターネットで提供されているコンテンツの閲覧などが行えるようになります。たとえば、スマホに対して提供されるネットワークがNAT64+DNS64を利用して運用されている場合、そのスマホを利用するユーザはIPv6しか使わなくなりますが、IPv4インターネットに接続されたWebサーバからコンテンツを取得できるようになります。ユーザが利用するホストに対してIPv6のみを提供しつつ、そのIPv6ネットワークからIPv4インターネットとの通信も行えるようにするための要素技術がNAT64+DNS64だといえるでしょう。

NAT64とDNS64を利用した単純なネットワーク構成を図23.3に示します。

IPv6ネットワーク内に接続された機器は、DNS64などの仕組みを利用して、IPv4アドレスが埋め込まれたIPv6アドレスを取得します。そして、NAT64機器が宛先IPv6アドレスから

[†13] 日本語では、**NAT64**を「なっと・ろくよん」と読みます。「なっと・ろくじゅうよん」とは読まないのでご注意ください。**DNS64**も「でぃーえぬえす・ろくよん」です。英語圏では、「NAT64」を「なっと・しっくす・ふぉー」、「DNS64」を「でぃーえぬえす・しっくす・ふぉー」と表現することが多いようです。

(cc) BY NC SA



▶図23.3　NAT64利用例

宛先IPv4アドレスを生成します。IPv4アドレスが埋め込まれたIPv6アドレスのフォーマットは、RFC 6052で規定されています（23.4節参照）。

iOSアプリのIPv6対応

エンジニアの中には、2016年6月からiOSアプリの審査基準としてIPv4に依存するコードの禁止が追加され、iOSアプリでIPv6対応が義務となったことから、IPv6に関する知識が必須になったという方も多いでしょう[†14]。Apple社のサイトには、「IPv4アドレス在庫枯渇の発生とともに、ユーザに対してIPv6のみによるインターネット接続性を提供するNAT64とDNS64がエンタープライズ網や携帯電話網で採用されることが増えている」とあります[†15]。

iOSアプリ開発者は、NAT64とDNS64環境でもアプリが正しく動作することを求められています。Appleのサイトでは、NAT64とDNS64はOS X 10.11から標準搭載されるようになっているので、Macを使ってiOSアプリの動作確認をすることを推奨しています。

23.3.1　ステートフルNAT64

RFC 6146で定義されているNAT64の主な想定環境は、「IPv6ネットワークに接続されたユーザから開始される通信がNAT64デバイスによって変換され、IPv4ネットワークに接続された機器と通信を行う」というものです。その際に、IPv6アドレスとIPv4アドレスの変換が行われますが、IPv4アドレスが貴重な資源なので、さまざまなIPv6アドレスが同一のIPv4アドレスを共有しながら利用することが想定されています。そのため、NAT64ではIPアドレス情報だけでの変換は行わず、TCPとUDPのポート番号を同時に利用したマッピングが主な利用方法になりそうです。

この場合、IPv6アドレスとTCP/UDPポート番号、IPv4アドレスとTCP/UDPポート番号のバインディング情報がNAT64機器によって保持されることになります。このようなNAT64の用法をステートフルNAT64とも呼ぶこともあります。

---

†14　Appleの発表：https://developer.apple.com/news/?id=05042016a

†15　“Apple Developer: Supporting IPv6 DNS64/NAT64 Networks”
https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/UnderstandingandPreparingfortheIPv6Transition/UnderstandingandPreparingfortheIPv6Transition.html

CC BY NC SA

ネットワーク層の情報だけではなくTCPやUDPのポート番号を活用したNAPTがほぼ前提となることから、NAT64機器ではIPv4におけるNAPT機器と同様の要求を満たす必要があります。具体的には、下記のようなRFCへの対応がRFC 6146によって求められています。

- RFC 4787[†16]：NATによるUDPの扱いを規定
- RFC 5382[†17]：NATによるTCPの扱いを規定
- RFC 5508[†18]：NATによるICMPの扱いを規定

また、ICE（Interactive Connectivity Establishment）など、NAT越え技術との互換性も求められています。ICEについては、RFC 5245[†19]をご覧ください。

23.3.2 NAT64におけるSIITの利用

IPv4とIPv6はまったく異なるプロトコルなので、IPv4-IPv6変換では、IPv4とIPv6プロトコルの差異を考慮しながらIPヘッダを変換する必要があります。たとえば、IPv4とIPv6とでフィールドの定義が異なる場合もあります。IPv4のTotal LengthフィールドはIPv4ヘッダとオプションを含む長さですが、IPv6のPayload LengthフィールドはIPv6ヘッダを含みません。

そこでNAT64では、IPv4ヘッダやIPv6ヘッダに含まれるフィールドを解釈して変換するために、両者の差異を吸収しつつ変換可能な方法を定義したSIIT（RFC 7915）を利用しています。

NOTE

RFC 7915ではICMPの扱いも定義されています。ただし、IPv4のICMPとIPv6のICMPv6は仕組みが異なるので、変換が不可能な場合もあります。また、IPv4ではIGMPであったものがIPv6ではICMPv6に含まれるなどの違いもあります。

SIITでは、IPとICMPのフィールドをIPv4とIPv6との間でどのように対応づけるかを定義しているだけです。そのため、IPv4におけるNATと同じ要領でIPv4-IPv6変換をしたい場合には、SIITの利用に加えて各セッションのステートを管理する手段が必要になります。NAT64ではそのための方法が規定されているといえます。

NAT64トランスレータ（SIITにおけるXLAT）は、IPv4のネットワークインターフェースとIPv6のネットワークインターフェースを持ちます。IPv4側から受け取ったパケットはIPv6へと変換され、IPv6側から受け取ったパケットはIPv4へと変換されます。NAT64、SIIT、DNS64を使うためのフレームワークを規定しているRFC 6144のシナリオ1によれば、変換に利用するIPv4側のアドレスはグローバルIPv4アドレス、IPv6側のアドレスはNAT64トランスレー

†16 RFC 4787：F. Audet, C. Jennings, “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP”, 2007年1月

†17 RFC 5382：S. Guha, K. Biswas, B. Ford, S. Sivakumar, P. Srisuresh, “NAT Behavioral Requirements for TCP”, 2008年10月

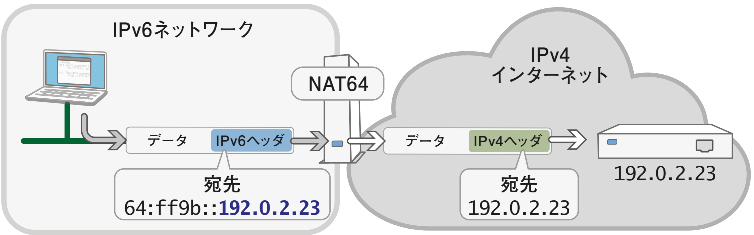
†18 RFC 5508：P. Srisuresh, B. Ford, S. Sivakumar, S. Guha, “NAT Behavioral Requirements for ICMP”, 2009年4月

†19 RFC 5245：J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols”, 2010年4月

CC BY NC SA

タ用のIPv6プレフィックスです。このNAT64トランスレータ用のIPv6プレフィックスとしては、RFC 6052に記載されている`64:ff9b::/96`がデフォルトで利用されます（他のIPv6プレフィックスを設定することもできます）。宛先アドレスがこのプレフィックスのIPv6パケットはNAT64トランスレータに転送され、他のIPv6アドレス宛のパケットは通常のIPv6ルータに転送されるようなネットワークを構築できます。

NAT64トランスレータは、IPヘッダに含まれるIPv4アドレスをIPv6アドレスに変換する必要があります。その変換方法はRFC 6052に規定されています。NAT64トランスレータ用のIPv6プレフィックス`64:ff9b::/96`が利用される場合、IPv6アドレスのインターフェース識別子となる残りの32ビットには、通信相手のIPv4アドレスが埋め込まれます。たとえば、`192.0.2.23`というグローバルIPv4アドレスを持つ相手と通信を行う場合、IPv6ネットワーク側では`192.0.2.23`が埋め込まれた`64:ff9b::192.0.2.23`というIPv6アドレスが利用されます。図23.4に、NAT64を利用した通信の例を示します。



▶ 図23.4　NAT64を利用した通信の例

23.3.3　DNS64

NAT64だけでは、ユーザ側の環境を変更することなくIPv6のみでIPv4インターネットの機器と通信できるようにはなりません。通信相手であるIPv4インターネットの機器の名前にAレコードだけしか存在しなければ、そもそもIPv6による通信を開始できないからです。IPv6対応済みのユーザ環境では、通常は通信相手の名前を解決する作業が行われ、その結果に応じて「IPv4で通信を行うのか、それともIPv6で通信を行うのか」を個々のアプリケーションが判断します。ユーザ側の環境がIPv6シングルスタックの場合、通信相手の名前にAAAAレコードが存在しなければ通信を開始できません。

状況を理解するために、名前解決が行われないIPアドレス直接指定の通信を考えてみましょう。たとえば、`http://192.0.2.23/`のようにIPv4アドレスがそのまま埋め込まれているURLでブラウザからアクセスしようとすると、IPv4アドレスが得られてしまいます。これではIPv6のみで運用されているネットワークからは通信を開始できないので、NAT64を経由してIPv4インターネットと通信を行うこともできません。

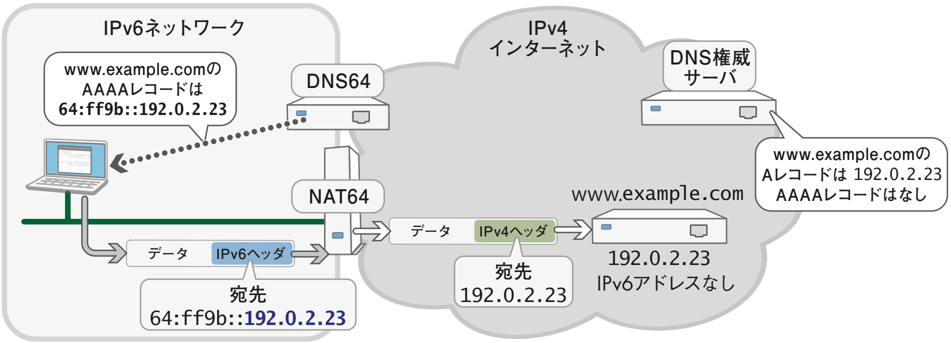
IPv4とIPv6は名前空間を共有しているので、DNSに細工をしつつNAT64を利用することで、IPv6のみを使いながらもIPv4インターネットと通信を行える仕組みが実現できるようになります。ユーザ側は、名前解決の結果AAAAレコードを得たことになり、IPv6での通信を開始するのですが、そのIPv6アドレスはNAT64ルータへと向かうように設定することができま

CC BY NC SA

す。そのDNSに対する細工が、RFC 6147で定義されているDNS64です。

DNS64では、IPv4アドレスを示すAレコードからIPv6を示すAAAAレコードを合成します。いくつかの制約はあるものの、NAT64とDNS64を組み合わせて利用することで、IPv6シングルスタックを利用しているユーザ機器で特別な変更を施さずに、IPv4インターネットにつながっている機器と通信できる環境を構築可能です。

図23.5に、NAT64とDNS64を利用した通信の例を示します。



▶ 図23.5　DNS64を利用した通信の例

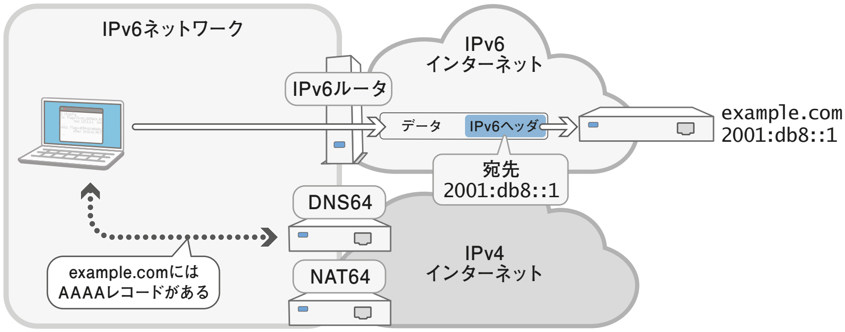
図23.5では、ユーザがIPv6のみで運用されるIPv6ネットワークに接続しており、キャッシュDNSサーバとしてDNS64機器を設定します。`example.com`はIPv4アドレス`192.0.2.23`のみで運用されているので、IPv6での通信は行えません。NAT64の環境でIPv6のみを利用するユーザが、IPv4インターネットに接続された機器と通信を行うには、DNS64を経由する名前解決が必要になります。

DNS64機器では、ユーザからのリクエストに応じて名前解決を行う際、IPv6アドレスを示すAAAAレコードがある場合にはその結果をそのまま返し、IPv4アドレスを示すAレコードしか存在しない場合にはRFC 6052で定義されている方法で「IPv4アドレスが埋め込まれたIPv6アドレス」を返します。ユーザが`example.com`の名前解決を行ったときにNAT64ルータを経由するIPv6アドレスをDNS64サーバが返すことで、IPv4で運用されたサーバへとユーザを誘導するためにNAT64ルータへとユーザのトラフィックが誘導されます。

NAT64ルータから送信されたIPv4パケットは、`example.com`へと送信され、`example.com`からの返信はNAT64ルータへと送信されます。IPv4パケットを受け取ったNAT64ルータは、IPv4パケットをIPv6パケットへと変換し、ユーザへと届けます。このようにして、NAT64を経由することで、IPv4のみで運用されているWebサイトをIPv6環境でブラウズできるようになります。

NOTE

DNS64は、NAT64におけるIPv4アドレスとIPv6アドレス変換規則と連携して動作する必要があります。IPv4アドレスから生成されるIPv6アドレスに利用されるプレフィックス情報は、DNS64とNAT64の間で同じものが利用される必要があります。DNS64とNAT64の間でいくつかの設定情報が共有される必要はありますが、DNS64とNAT64の間で個別セッションに関するステートは共有されません。そのため、動作としては、DNS64とNAT64は独立しています。



▶ 図 23.6　IPv6 トラフィックはそのまま

23.3.4　NAT64+DNS64 を利用した IPv6 シングルスタック運用

NAT64+DNS64 の特長は、IPv6 のシングルスタックネットワーク環境から IPv4 インターネットへの通信を可能にしつつ、IPv6 インターネットとの間ではそのまま IPv6 インターネットで通信可能になることです。NAT64 を利用して IPv4 インターネットと通信できる IPv6 ネットワークから、IPv6 インターネットとの通信を行う場合は、図 23.6 のようにそのまま IPv6 で通信できます。

RFC 6586[†20]では、それまで IPv4 と IPv6 のデュアルスタックで運用されていたネットワークを IPv6 シングルスタックへと変更し、少ないユーザをその環境に移動させた実例が要約されています。NAT64 と DNS64 を利用して IPv4 インターネットへの接続性を確保しつつ、ユーザに対して IPv6 シングルスタックサービスを提供した経験を文書化したものといえるでしょう。IPv6 対応していないアプリケーションや、IPv6 環境で実行されたことがないコードのバグへの対処といった課題はあるものの、IPv6 シングルスタック環境であっても概ね問題なくユーザが利用できたと結論づけられています。

IPv4 と IPv6 のデュアルスタック環境では、IPv6 での接続をアプリケーションが試みて失敗し、IPv4 での接続を試みるというフォールバックが発生することがあります。NAT64 と DNS64 で端末を IPv6 シングルスタックにすると、IPv6 から IPv4 へのフォールバックが発生しないという効果もあります。

一方で課題も少なくありません。RFC 6586 の環境では、23.5 節で説明する ALG として RFC 6384 にある FTP 以外を提供しなかったところ、インスタントメッセージサービスや VoIP アプリケーションなど、プロトコル中で IPv4 アドレス情報を運ぶアプリケーションが動作しなくなったとあります。インターネットを利用するゲームの多くも利用不能だったようです。その他、ファイアウォールの IPv6 対応が不十分であるという課題にも直面したと記述されています。

HTML に IPv4 アドレスが直接記述されているような場合、DNS が利用されないので DNS64 による変換が行われず、Web ページの一部が見られなくなる可能性についても紹介されています。NAT64 と DNS64 による IPv6 シングルスタックへの移行は、用途を選んで問題が発生しな

†20　RFC 6586：J. Arkko, A. Keranen, “Experiences from an IPv6-Only Network”, 2012 年 4 月

cc BY NC SA

い範囲であればサービスの提供が可能である、というのが実状なのかもしれません。

Web サーバで NAT64+DNS64 に対応するために必要なこと

HTML のページ内に IPv4 アドレスがハードコーディングされたリンクがあるような状況では、DNS64 が使えません。Web サーバ管理者は、Web サーバに含まれるコンテンツが DNS64 に対応していることを確認する必要があります。

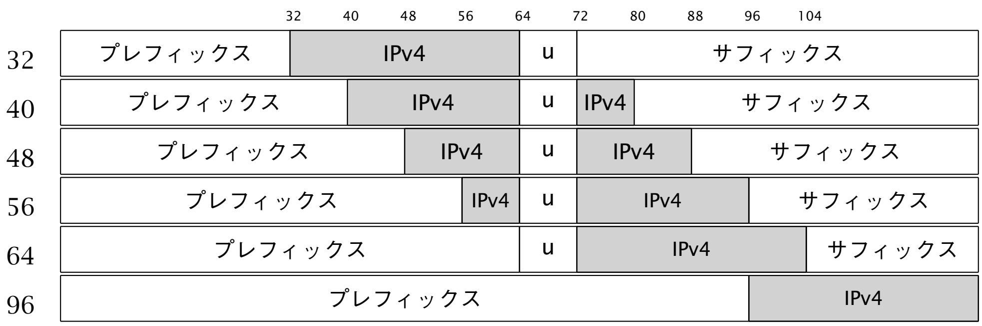
それ以外には、NAT64+DNS64 を利用しているユーザとの通信を行うために、Web サーバ側で何か特別なことをする必要はありません。NAT64+DNS64 は、IPv4 で運営されている機器に対して IPv6 オンリーの環境から通信できるようにするための仕組みであり、IPv4 で運営されている Web サーバが何もしなくても通信できるようにするための仕組みなのです。

ただし、IPv4 と IPv6 は直接の互換性がない異なるプロトコルであり、環境や設定によっては NAT64+DNS64 を利用した通信を実現できない場合もありえます。そもそも NAT64+DNS64 は NAT の一種なので、従来の NAT が抱える問題点をそのまま抱えています。また、フラグメンテーションに関連するパケットや、IPv6 ヘッダオプションが付属したパケットなど、IPv4 と IPv6 のプロトコルとしての差異が原因で変換できないパケットもあります。

そうした問題に対する回避策として考えられるのは、Web サーバ側の IPv6 対応です。NAT64+DNS64 はあくまでも IPv6 で通信できない相手と通信するための手段であり、IPv6 と通信できる相手との通信であれば、NAT64 を経由せずに IPv6 でユーザと直接通信できます。

23.4 IPv4/IPv6 変換機用 IPv6 アドレス

IPv4/IPv6 変換機で利用するための IPv6 アドレスフォーマットは RFC 6052 で定義されています。RFC 6052 では、プレフィックス長が 32、40、48、56、64、96 の 6 種類の場合について、図 23.7 のようにフォーマットが定義されています。それぞれの IPv6 アドレスの中に IPv4 アドレスが埋め込まれることで、IPv4 と IPv6 アドレスの変換をステートレスに行えるフォーマットになっています。



▶ 図 23.7　IPv4/IPv6 変換機用 IPv6 アドレスフォーマット

図23.7のuが示している64ビットめから71ビットめは、RFC 4291[†21]で定義されているホスト識別子フォーマットとの互換性のために予約されており、すべてゼロにセットされます。

RFC 6052では、IPv4/IPv6変換機専用のIPv6アドレスプレフィックスも定義されています。具体的には、`64:ff9b::/96`というプレフィックスに対して、下位32ビットをIPv4アドレスとします。このIPv6アドレスは、IPv6アドレスのチェックサムがIPv4アドレスと同じになるようになっています。先頭16ビットの`0064`と、17ビットめから32ビットめまでの`ff9b`のチェックサムはゼロになります。

23.4.1　ローカルな管理ドメイン内で使用するプレフィックス

RFC 8215[†22]では、ローカルな管理ドメイン内でIPv6/IPv4変換を利用する際に利用可能なプレフィックスとして、`64:ff9b:1::/48`が予約されています。下位32ビットにIPv4アドレスを埋め込むという使い方は、RFC 6052の`64:ff9b::/96`と同じです。

RFC 6052で予約された`64:ff9b::/96`は、プライベートIPv4アドレスへの変換を行うための用途が禁止されているため、IPv6からプライベートIPv4アドレスへ変換するときには利用できませんでした。RFC 8215では、そういった用途での利用も可能なIPv6プレフィックスを予約しています。

RFC 8215で定義されている`64:ff9b:1::/48`は、RFC 6052で定義されている`64:ff9b::/96`のようなチェックサム中立性がありません。

23.5　ALG

IPv4のNAT同様に、IPv4/IPv6変換においても**ALG**（Application Layer Gateway）は重要な要素です。ペイロード中にIPアドレス情報が含まれているプロトコルは、ALG機能がなければ通信を正常に行えません。

本書執筆時点でRFC化されているのは、FTPのALGのみとなっています。

23.5.1　IPv6からIPv4への変換用FTP ALG

IPv6からIPv4への変換用のFTP ALGは、RFC 6384で定義されています。

FTPそのものは、RFC 959[†23]で定義されている古いプロトコルです。FTPには、サーバがクライアントに対して接続を張るアクティブモードと、サーバが新たなポートを開いてクライアントを待つパッシブモードの、2つのモードがあります。アクティブモードとパッシブモードの両方とも、IPv4アドレスがコマンド中に登場します。アクティブモードのPORTコマンドの引数にIPv4アドレスが含まれ、パッシブモードのPASVコマンドに対するサーバ側回答にIPv4アドレスが含まれます。

このように、RFC 959におけるFTPコマンドはIPv4に限定され、IPv6では利用できません。そのため、RFC 2428[†24]でFTPのIPv6拡張が定義されています。そこで定義されているのは、

---

†21　RFC 4291：R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, 2006年2月
†22　RFC 8215：T. Anderson, “Local-Use IPv4/IPv6 Translation Prefix”, 2017年8月
†23　RFC 959：J. Postel, J. Reynolds, “File Transfer Protocol”, 1985年10月
†24　RFC 2428：M. Allman, S. Ostermann, C. Metz, “FTP Extensions for IPv6 and NATs”, 1998年9月

アクティブモード用のEPRT（Extended Port）コマンドと、パッシブモード用のEPSVコマンドです。これらのコマンドは、IPv4とIPv6の両方で利用可能です。

IPv6でFTPを利用する場合、EPSVもしくはEPRTコマンドを利用することになりますが、IPv4のみで運用されている既存のすべてのFTPサーバがそれらに対応しているわけではありません。対応はしていても挙動がおかしいFTPサーバもあるので、RFC 6384では、EPSVからPASVの変換、EPRTからPORTへの変換手法が定義されています。

RFC 6384では、FTP ALGの設定や状態を知るためのALGSコマンドの実装も要求しています。ALGSコマンドにより、EPSVからPASVへの変換をFTP ALGで行うべきであるかどうかを設定可能です。また、設定されている変換内容を確認できます。

なお、RFC 6384では、セキュリティ上の懸念として、暗号化しないFTPを使うことの危険性について注意されています。FTPにはTLSによる暗号化の仕組みもありますが、ALGがそれに対応しており、かつ、ALGが介在しても通信が可能な方式でなければ、ALG経由で利用できるのは暗号化しない方式だけになります。

23.5.2 TURN

RFC 5766で定義されているTURNは、IPv4のみ対応しており、IPv6は範疇外です。IPv6に対応したTURNは、RFC 5776に対する拡張として、RFC 6156で別途定義されています。

RFC 5766のTURNは、IPv4とIPv4での通信のみを対象としていますが、RFC 6156はIPv4とIPv6の間でのリレーも対象としています。このため、RFC 6156の拡張によって、IPv4ノードとIPv6ノードの通信を行うためのALGとしてTURNを利用することもできます。

TURNに関しては、20.3節を参照してください。

CC BY NC SA

[書籍版では空白のページです]

BY NC SA

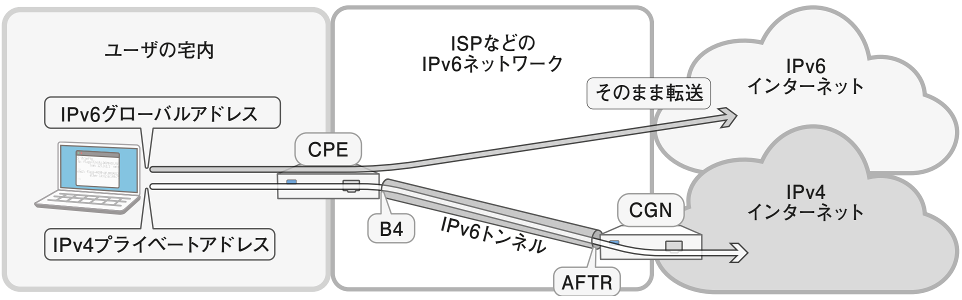
第24章

IPv4/IPv6共存技術の運用形態

さまざまな技術を組み合わせることでIPv4/IPv6共存技術を実現する手法がいろいろ提案されています。これらの技術は、環境や事業者の運用形態に応じて最適なものが決まるため、それぞれ優劣をつけるというよりも、各自にあったものを選択することになるでしょう。

24.1 DS-Lite

DS-Lite（Dual-Stack Lite）は、基幹ネットワークをIPv6だけで構築したうえでグローバルIPv4アドレスを節約する手法です。RFC 6333[†1]で規定されています。18.4.1項で説明したCGN（Carrier Grade NAT）を利用したNAT444と違い、NATが1段で済む点がDS-Liteの特徴です。



▶ 図24.1　DS-Lite概要

DS-Liteでは、ISPなどのバックボーンネットワークをIPv6のみで構成し、家庭内などの利用者の環境にはDS-Lite対応CPE（Customer Premises Equipment）を設置します。そして、家庭内などでは、IPv6グローバルアドレスとIPv4プライベートアドレスを利用します。

CPEは、IPv6パケットをそのままバックボーンネットワークへと転送します。IPv4パケットは、CPEによってIPv6カプセル化されて、バックボーンネットワークに設置されたAFTR（Address Family Transition Router）と呼ばれる機器まで転送されます。AFTRまでのカプセ

†1　RFC 6333 : A. Durand, R. Droms, J. Woodyatt, Y. Lee, “Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion”, 2011年8月

cc BY NC SA

ル化を行う機能はB4（Basic Bridging BroadBand）と呼ばれます。AFTRは、CPEによってカプセル化されたIPv4パケットをIPv6パケットから取り出したうえでNATによる変換を行い、IPv4インターネットへとパケットを転送します。

NOTE

B4はB-fourと発音し、「前」という意味の英単語「Before」と同じように発音されます。一方、AFTRは「後」という意味を持つ英単語「After」から母音のeを抜いた形になっています。

DS-Liteの利点として、バックボーンネットワークをシングルスタックで運用できることから、デュアルスタック運用にかかるコストを削減できるという点が挙げられます。

DS-LiteとDOCSIS 3.0

DS-Liteの提案者の中心であるAlain Durand氏は、提案時点ではCATV網を運用するComcastに所属していました（RFC 6333発行時点ではJuniper Networks）。それもあって、DS-Liteの仕組みは、CATV網の技術と相性が良いものになっています。

CATV網では、IPを利用した通信のために、各CPEにIPアドレスを割り当てます。IPv4のプライベートアドレス10.0.0.0/8では1600万個のCPEまでしか制御できませんが、これではComcastのような大規模なユーザを抱えるCATV事業者には心もとないといえます。そこで、既存のCATV網上で高速な通信を行うための国際規格であるDOCSISでは、2006年に改訂されたバージョン3.0からIPv6に対応しています。DOCSIS 3.0により、CATV事業者は、自社の網内をIPv6だけで構成できるようになりました。

さらにDS-Liteのような仕組みを導入することで、ユーザに対するIPv4インターネットへの接続性は、そのIPv6上に構築したトンネル経由で提供できます。DS-Liteが米国最大のCATV事業者であるComcast社員によって提案されたのは、実に自然なことだったといえるでしょう。

24.2　Lightweight 4over6（lw4o6）

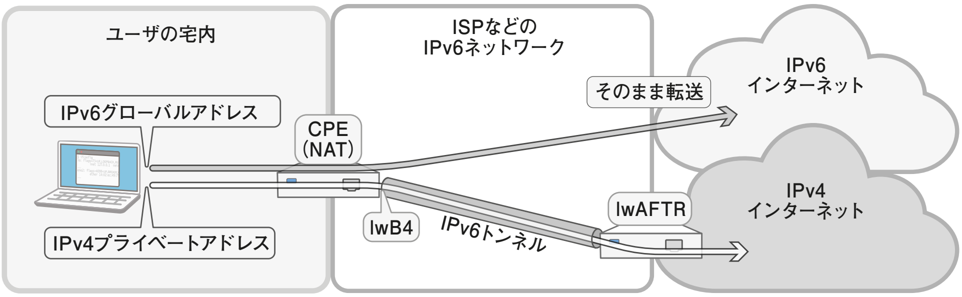
DS-Liteに対して拡張を加えた**Lightweight 4over6**（**lw4o6**）と呼ばれる規格がRFC 7596[†2]として定義されています。lw4o6では、DS-Liteでは中央で管理していたNAPTの機能をユーザの手もとにあるCPEに移すことで、運用者がCGNを運用せずに済むようにしています。DS-Liteは、デュアルスタックを手軽に実現できることを連想させる名称でしたが、実際にはCGNの運用にかかる部分が手軽とはいえなかったので、その部分を軽減した標準がlw4o6であるといえます。

当然ですが、lw4o6のアーキテクチャはDS-Liteに似ています。lw4o6でも、DS-Liteと同様に、IPv4を運ぶためにIPv6トンネルを利用します。また、IPv6トンネルの両端となるB4とAFTRという構成要素もDS-Liteと同じです。RFC 7596では、lw4o6拡張のあるB4をlwB4、

†2　RFC 7596：Y. Cui,Q. Sun,M. Boucadair,T. Tsou,Y. Lee,I. Farrer, “Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture”, 2015年7月

CC BY NC SA

AFTRをlwAFTRと呼んでいます。



▶ 図24.2 Lightweight 4over6概要

DS-Liteとlw4o6の大きな違いは、DS-LiteではNAPT機能がAFTRに入る一方で、lw4o6ではlwB4に入るという点です。DS-LiteではAFTRが担っていたNAPT機能をlwB4が担うので、NAPTに必要となるパブリックIPv4アドレスとポート番号のセットをlwB4が知っている必要があります。lwB4は、RFC 7598[†3]にあるSoftwire46のDHCPv6オプションを受け取ることで、それらの情報を得ます。

lwB4は、受け取った情報からlw4o6のIPv6アドレスを生成します。IPv6アドレスの生成方法は、RFC 7597[†4]で定義されているMAP-Eにおける手法を利用します。MAP-EにおけるIPv6アドレスの生成については24.4.1項を参照してください。

24.3 A+P

A+P（**Address plus Port**）は、その名前のとおり、IPv4アドレスとTCP/UDPポート番号のセットを利用することでIPv4アドレスの共有を行う手法です。具体的には、複数のCPEが同じIPv4アドレスを利用しつつ、割り振られたTCP/UDPポート番号のみを利用します。CPEからのIPv4パケットは、PRR（Port-Range Router）と呼ばれるルータまでトンネルを通って転送されます。IPv4インターネット側からのパケットは、PRRがTCP/UDPポート番号に応じて適切なCPEとつながるトンネルへと転送します。

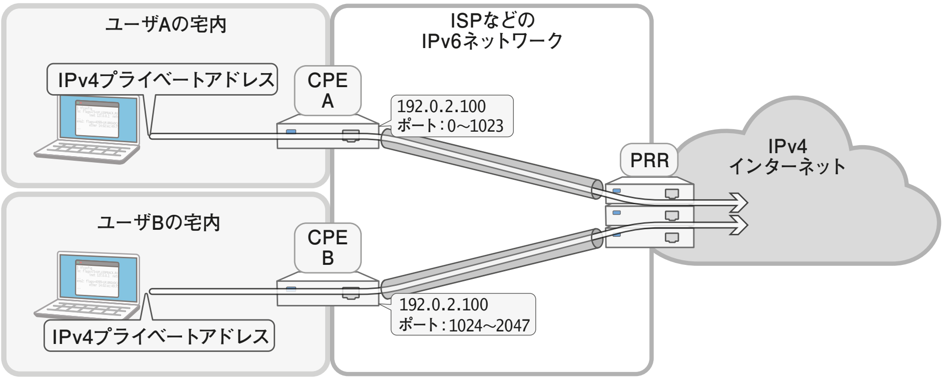
図24.3では、PRRが`192.0.2.100`というグローバルIPv4アドレスを利用しています。CPE AとCPE Bも、PRRと同じ`192.0.2.100`というIPv4アドレスを利用しますが、CPE Aが利用するTCP/UDPポート番号が0〜1023、CPE Bが利用するTCP/UDPポート番号が1024〜2047という違いがあります。

各CPEは、PRRへとパケットを送信する前に、必要に応じてNATやポート番号を変換します。このとき、各CPEは、割り当てられたTCP/UDPポート番号の範囲内であれば、NATによる変換をせずにパケットを転送することも可能です。たとえば、`192.0.2.100`というIPv4アドレスを家庭内のPCが利用していて、TCP/UDPの送信元ポート番号をCPEに割り当てられた

†3 RFC 7598 : T. Mrugalski,O. Troan,I. Farrer,S. Perreault,W. Dec,C. Bao,L. Yeh,X. Deng, “DHCPv6 Options for Configuration of Softwire Address and Port-Mapped Clients”, 2015年7月

†4 RFC 7597 : O. Troan, W. Dec, X. Li, C. Bao, S. Matsushima, T. Murakami, T. Taylor, “Mapping of Address and Port with Encapsulation (MAP-E)”, 2015年7月

CC BY NC SA



▶ 図24.3　A+Pの構成例

ものの範囲内に限定できるのであれば、CPEでNATによる変換をせずに通信が可能です。

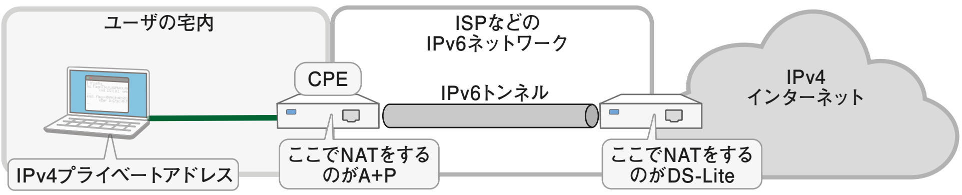
24.3.1　DS-Liteとの関係

DS-LiteとA+Pは、技術的な観点ではよく似た仕組みです。実際、DS-LiteとA+PがInternet Draftとして議論されていたとき、両方を統合するという案もありました。しかし、最終的にDS-LiteはStandard TrackでRFCになり、A+PはExperimentalなRFC 6346[†5]になりました。

技術的には似た面もありますが、A+PとDS-Liteとでは、提案のモチベーションが大きく異なります。DS-Liteのモチベーションは、デュアルスタック運用のコスト削減です。これに対し、A+Pのモチベーションは、CGNの問題を解消することにあります。A+PのRFCでは、CGNの問題として、UPnPが使えない、多くのユーザの通信ステートをCGNで保持しなければならない、単一障害点になりうるといった点が具体的に挙げられています。

DS-LiteとA+Pでは、提案の方針にも大きな違いがあります。DS-Liteでは具体的な技術が述べられているのに対し、A+Pでは考え方が中心に述べられているのです。たとえば、DS-LiteではB4とAFTRの間のトンネルをIPv6で構築します。一方、A+Pでは、L2延伸を含む何らかの方法で行うとあります。A+Pで必要となるシグナリングプロトコルも、詳細がRFCに書かれているわけではなく、可能性がある手法がいくつか列挙されています。

NATによる変換が発生する場所もDS-LiteとA+Pでは異なります。図24.4のように、DS-LiteがIPv4ネットワークの手前でNATを行うのに対して、A+PはCPE側でNATを行えます。さらに、A+PではNATを使わないことも可能です。



▶ 図24.4　DS-LiteとA+Pの違い

†5　RFC 6346：R. Bush, "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", 2011年8月

CC BY NC SA

ステートレスであることもA+Pの大きな特長です。A+Pでは、グローバルIPv4アドレスで利用するUDP/TCPのポート番号をNATで利用するので、ステートを保持せずに自動的な変換が可能になっています（ただし、ICMPやフラグメントされたIPパケットについては、通常のNAT同様にステートフルな変換が必要です）。

24.4 MAP-E、MAP-T、4rd

IPv6ネットワークを通じてIPv4サービスを提供するための方法として、**MAP-E**、**MAP-T**、**4rd**というものがあります。この3つは、IETFのsoftwireワーキンググループで同時期に議論が続けられていたプロトコルです。それぞれ、以下のRFCとして発行されています。

- MAP-E ： RFC 7597
- MAP-T ： RFC 7599[†6]
- 4rd ： RFC 7600[†7]

上記のうち、RFC 7597（MAP-E）とRFC 7599（MAP-T）がStandard Track、RFC 7600（4rd）がExperimentalです。

MAP-EとMAP-Tは、MAP（Mapping of Address and Port）と呼ばれる仕組みが2つの異なる方式に分かれたものです。MAP-T[†8]は、Mapping of Address and Port with Translationであり、途中でトランスレーションを行います。MAP-E[†9]は、Mapping of Address and Port with Encapsulationであり、途中でカプセル化が行われます。

4rdは、時期によって異なる仕組みを指すので注意が必要です。MAPがSAM（Stateless Address Mapping）と呼ばれるステートレスなトンネリングプロトコルから進化する過程では、さまざまな個人が4rdという名称のプロトコルを提案するInternet Draftを発表しました。最終的に、6rdを提案したRemi Despres氏らによって提案された4rd-U[†10]と呼ばれる仕組みが4rdと改称され[†11]、これがExperimentalなRFC 7600として発行されました。

24.4.1 MAP-TとMAP-E

MAP-EおよびMAP-Tは図24.5のような構成で利用されます。

MAP-EとMAP-Tの主な違いは、IPv4パケットをIPv6ネットワークで転送する方式です。MAP-Eでは、RFC 2473にある方法でIPv4パケットをIPv6でカプセル化します。一方、MAP-T

†6 RFC 7599：X. Li, C. Bao, W. Dec, O. Troan, S. Matsushima, T. Murakami, “Mapping of Address and Port using Translation (MAP-T)”, 2015年7月

†7 RFC 7600：R. Despres, S. Jiang, R. Penno, Y. Lee, G. Chen, M. Chen, “IPv4 Residual Deployment via IPv6 - A Stateless Solution (4rd)”, 2015年7月

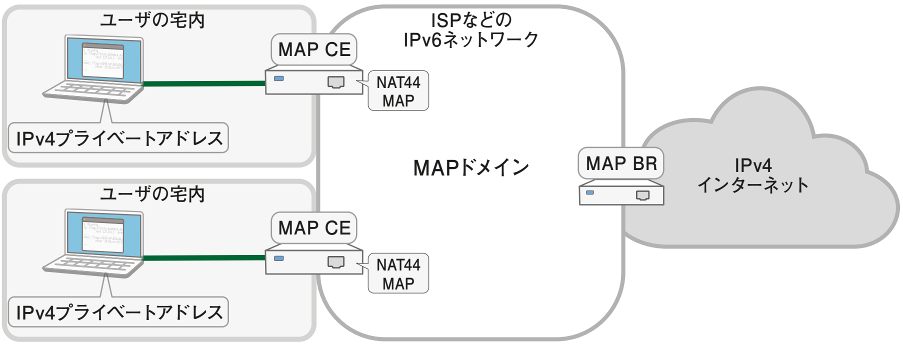
†8 X.Li, C.Bao, W.Dec, O.Troan, S.Matsushima, T.Murakami, “Mapping of Address and Port using Translation (MAP-T)”, 2012年10月：`https://datatracker.ietf.org/doc/draft-ietf-softwire-map-t/`

†9 W.Dec, X.Li, C.Bao, S.Matsushima, T.Murakami, “Mapping of Address and Port with Encapsulation (MAP)”, 2012年9月：`https://datatracker.ietf.org/doc/draft-ietf-softwire-map/`

†10 R.Despres, Ed, R.Penno, Y.Lee, G.Chen, J.Qin, “IPv4 Residual Deployment via IPv6 - Unified Solution (4rd)”, 2012年3月：`https://datatracker.ietf.org/doc/draft-despres-softwire-4rd-u/`

†11 S.Jiang, R.Despres, R.Penno, Y.Lee, G.Chen, M.Chen, “IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd)”, 2012年10月：`https://datatracker.ietf.org/doc/draft-ietf-softwire-4rd/`

CC BY NC SA



▶ 図24.5　MAP-E および MAP-T の概要

では、SIIT（RFC 7915）を利用して変換を2回行います。

MAP-E および MAP-T では、MAP CE（Customer Edge）と MAP BR（Border Relay）と呼ばれる装置を利用します。MAP CE は、IPv6 で構成された MAP ドメイン内に利用者のネットワークを接続するもので、一般には CPE（Customer Premises Equipment）や RG（Residential Gateway）などと呼ばれている機器ですが、MAP-E および MAP-T の文書ではこれが「CE」と表現されています。MAP BR は、MAP ドメイン外の IPv4 アドレスを宛先とするときに利用されます。

MAP CE は、ユーザからの IPv4 パケットの宛先を確認し、それが MAP ドメイン内であるかどうかを判断します。MAP ドメイン外の IPv4 アドレスを宛先とするパケットである場合、そのパケットは MAP BR の IPv6 アドレスを宛先として MAP ドメイン内の IPv6 ネットワークで送信されます。このとき、MAP-E であれば IPv4 パケットは IPv6 でカプセル化され、MAP-T であれば IPv4 ヘッダが IPv6 ヘッダへと変換されたうえで送信されます。

MAP CE は、ユーザからの IPv4 パケットの宛先が MAP ドメイン内のものであれば、宛先 IPv4 アドレスと TCP もしくは UDP ポート番号から宛先 IPv6 アドレスを生成します。このとき、トランスポート層プロトコルのポート番号も利用することで、同じ IPv4 アドレスを複数の MAP CE で共有できるようにしています。したがって、図24.5 の NAT44 は、その MAP CE に割り当てられたポート番号の範囲内となるように変換する機能を備えた NAT 機能を示しています。

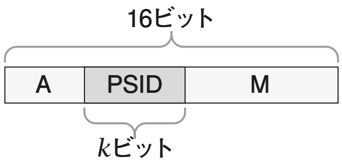
同じようにポート番号を利用する A+P では、B4 と AFTR の間のトンネルを構築する手法が明記されていません。一方、MAP-E および MAP-T では、MAP CE と MAP BR の間の通信を IPv6 で行うことが前提となっています。さらに、IPv4 アドレスとポート番号を IPv6 アドレスにマッピングする手法にも工夫が施されており、IPv4 アドレスとポート番号から自動的に MAP ドメイン内の宛先 IPv6 アドレスを計算可能です。逆に、IPv6 アドレスから、各 MAP CE に対応する IPv4 アドレスとポート番号群を自動的に計算することもできます。

各 MAP CE では、利用するポート番号群を、PSID（Port-Set Identifiers）と呼ばれる識別子からアルゴリズムで決定します。PSID からポート番号をマッピングするアルゴリズムは、GMA（Generalized Modulus Algorithm）と呼ばれ、“Extended IPv6 Addressing for Encoding Port

[CC BY-NC-SA]

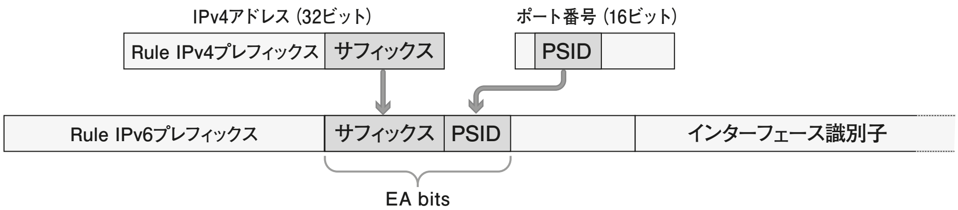
Range" と題された Internet Draft[†12] で説明されています。PSID は各 MAP CE で設定され、16 ビットのポート番号のうち一部のマスク値として運用されます。ある MAP CE が扱えるポート番号は、この PSID をその一部に含むようなものになります。連続したポート番号でも飛び飛びのポート番号でも表現可能です。MAP CE は基本的に同じ PSID を使い続けるので、事実上、PSID が MAP CE に割り当てられたポート番号を示すことになります。

図 24.6 に、16 ビットのポート番号に *k* ビットの PSID が含まれている状態を示します。各 MAP CE が利用できるポート番号は、PSID の部分を固定値とし、A および M の部分を任意の値とするようなものになります。ただし、IANA によってシステムポートとして割り当てられている番号を利用してしまわないように、A の部分の長さが 0 でないときは A の値を 0 よりも大きくする必要があります。



▶ 図 24.6 MAP-E および MAP-T の PSID

MAP ドメイン内では、各 MAP CE に対し、IPv4 アドレスと PSID が割り当てられます。それらの情報を利用して、MAP CE の IPv6 アドレスを生成します。具体的には、図 24.7 のように、IPv6 プレフィックス部分に IPv4 アドレスのホスト部と PSID が埋め込まれます。この埋め込まれた部分は EA bits と呼ばれます。EA bits の長さと、MAP を利用して転送することを示す IPv4 および IPv6 のプレフィックス（Rule IPv4 プレフィックスおよび Rule IPv6 プレフィックス）は、あらかじめ MAP CE に設定されます。



▶ 図 24.7 MAP-E および MAP-T での IPv6 アドレス生成

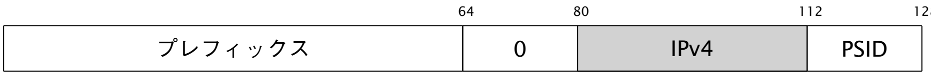
たとえば、EA bits の長さが 16 ビット、Rule IPv6 プレフィックスが `2001:db8:0000::/40`、Rule IPv4 プレフィックスが `192.0.2.0/24`、IPv6 のプレフィックスが `2001:db8:0012:3400::/56` に設定されている MAP CE があるとします。Rule IPv4 プレフィックスの長さが 24 ビットなので、図 24.7 のサフィックスの部分の長さは 8 ビットと決まります。そして、`2001:db8:0012:3400::/56` におけるサフィックスの部分（先頭の 40 ビットに続く 8 ビット）は

†12 https://tools.ietf.org/html/draft-bcx-behave-address-fmt-extension-11（本書執筆時点）

CC BY NC SA

十進表記で18となるので、IPv4アドレスは192.0.2.18です。さらに、EA bitsの長さが16ビットなので、サフィックスの8ビットを除くと、PSIDが8ビットに決まります。2001:db8:0012:3400::/56におけるPSIDの部分（サフィックスに続く8ビット）は十六進表記で34なので、PSIDは34とわかり、ここからGMAによりポート番号群が計算できます。

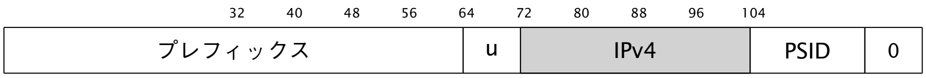
各MAP CEは、MAPドメイン内でIPv4パケットを転送する際に、IPv4アドレスとトランスポートプロトコルのポート番号さえあれば、PSIDを利用して宛先IPv6アドレスも自動生成できます。MAP-EおよびMAP-Tでは、このときのインターフェース識別子の生成方法が図24.8のように定義されています。このインターフェース識別子は、図23.7に示したRFC 6052のプレフィックス長が64ビットである場合のフォーマットと互換性があります。



▶ 図24.8　MAP-EおよびMAP-Tで利用されるインターフェース識別子

NOTE

MAPにおけるインターフェース識別子の生成方法は、RFC 7597になる前のdraft段階であるdraft-ietf-softwire-map-03からdraft-ietf-softwire-map-04で変わっています。日本国内で使われるMAP-E実装では、draft-03段階でのインターフェース識別子が使われることがあります。draft-03版のインターフェース識別子生成方法は、図24.9のように定義されています。このインターフェース識別子は、図23.7に示したRFC 6052のプレフィックス長が64ビットである場合のフォーマットと互換性があります。



▶ 図24.9　MAP-EおよびMAP-Tで利用されるインターフェース識別子（draft-03版）

MAP-Eで、ポート番号によるIPv4アドレス共有機能を使わない、つまりMAP CEでのNAT44機能を無効化すると、DS-LiteのAFTRと互換性があるプロトコルになります。NAT44機能が無効化されたMAP CEで、MAP BRとしてDS-Lite AFTRのIPv6アドレスを設定することで、DS-Lite AFTRを通じてIPv4インターネットとの通信が行えます。

MAPでは、MAP CEおよびMAP BRが自身が使うべきポートマッピングのルールやPSIDを何らかの方法で取得し、設定されることが前提になります。MAP CEに対して各種設定情報を伝えるための標準として、softwireワーキンググループにてMAP用のDHCPv6オプションの標準化が行われ、RFC 7598として発行されています。なお、このRFC 7598は、MAPだけではなくIPv4をIPv6ネットワークを通じて提供するための一般的なDHCPv6オプションを定義するものです。

24.4.2　MAP-TとMAP-Eの統合に関するコイントス

MAP-E、MAP-T、4rdという3つのプロトコルが乱立している状態については、IETFのsoftwireワーキンググループでも議論が続いていました。似て非なるプロトコルであるMAP-EとMAP-Tを1つの統合されたプロトコルとするのか、それとも別々のプロトコルとするのかに

ついては、2012年8月に行われたIETF84にて別々のプロトコルとすると決定したのですが、その決定にあたってはコイントスという異例の手法が採用されました。IETF84 softwireワーキンググループでの挙手による賛否の確認では双方の意見ともにほぼ同数だったこと、プロトコルの優劣とは違う話であったことから、コイントスが採用されたようです。

IETFでコイントスによってプロトコルの方向性が決定したという事例は、筆者もその知人も聞いたことがないので、もしかしたら初の試みだったのかもしれません。珍しい試みだったこともあって当時の風景を数人が撮影しており、ネット上に動画も投稿されています[†13]。

このコイントスは、唐突に行われたわけではなく、実際はその前に相応の動きがあったようです。softwireワーキンググループのチェアはIETF84の直前に交替しており、その直前にメーリングリスト上でMAP-EとMAP-Tに関する投票を行っていたのですが、それをまとめられなかったことや、それまでの議論のドライブがあまり上手に進行できなかったことなどが背景にあると噂されています。そういった経緯で登場した新しいsoftwireワーキンググループチェアは、延々と続いていたMAP-EとMAP-Tの議論に大鉈をふるうべく、コイントスという方法で一気に決定したようです。

IETF84で大鉈がふるわれたのは、MAP-EとMAP-Tの統合の賛否だけではありません。MAP-E、MAP-T、4rdのうちどれをStandard Trackにするのかという議論も、このIETF84で決着しました。softwireワーキンググループ参加者による挙手の結果、最終的にMAP-Eが多数の支持を得てStandard Trackになり、MAP-Tと4rdがExperimentalとなっています。カプセル化を行うMAP-Eと、IPv4からIPv6へと変換しその後IPv6からIPv4への変換を行うMAP-T、各種工夫によってIPv4パケットの情報が欠落しないようにしてある4rdを比べると、MAP-Eが最も素直でシンプルなプロトコルといえることから、多数決の結果としては妥当なものと考えられます。

ただし、その後のsoftwireワーキンググループでMAP-TもStandard Trackにすることが決定され、MAP-EのRFC 7597とMAP-TのRFC 7599が両方ともStandard Track、4rdのRFC 7600だけがExperimentalなRFCとして発行されることになりました。

24.4.3 4rd

IPv4とIPv6の間には互換性がなく、IPヘッダに含まれるフィールドの意味がIPv4とIPv6で異なっていたり、IPv4ヘッダにあったフィールドがなくなっていたり、新しくIPv6で登場したフィールドがあったりします。**4rd**では、IPv4とIPv6の差に伴う情報の欠落ができるだけ発生しないように、MAP-EやMAP-Tでは考慮されていないさまざまな工夫が施されています。一方で、仕組みが複雑化しているというトレードオフもあります。

4rdでは、IPv6拡張ヘッダを利用して、元のIPv4ヘッダに含まれる情報を埋め込む仕組みが提案されています。それ以外にも、フラグメンテーションやICMPの扱いなどで細かい工夫がいろいろ施されています。4rdにおけるIPv6拡張ヘッダの活用方法について詳しくはRFC 7600を参照してください。

[†13] Mark Townsley, “Rough consensus, running code, and the occasional coin toss”, 2012年8月：`https://www.facebook.com/photo.php?v=10150963959567666`

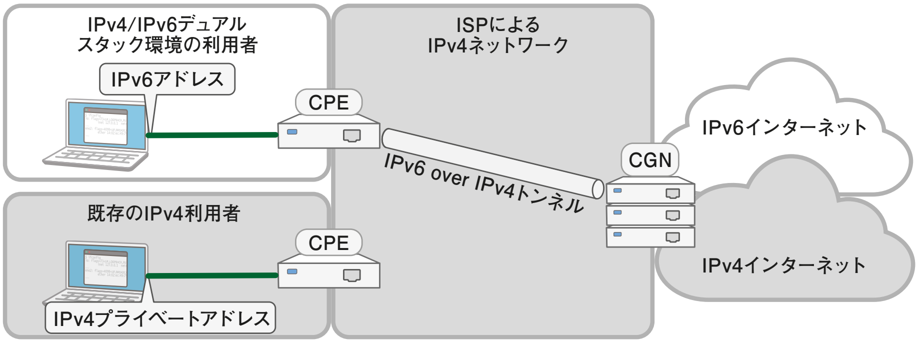
CC BY NC SA

24.5 CGNを利用して徐々にIPv6対応していく方法

第IV部で詳しく触れたIPv4アドレスの枯渇に伴い、ISPでは複数のユーザがIPv4アドレスを共有しながら使う環境を検討する必要が出てきます。そのために導入されるのがCGN（Carrier Grade NAT）です。つまり、CGNは、IPv6対応とは別の問題への対策としても必要になる設備です。

一方、多くの通信事業者にとっては、IPv4枯渇だけでなくIPv6対応も同時に進めるべき課題です。とはいえ、ネットワーク内を一気にIPv6対応することは現実的ではありません。そこで、少しずつ徐々にIPv6対応していくような漸進的手法が好まれます。

そのような背景から、CGNを導入しつつ既存利用者に影響を与えずに漸進的なIPv6対応を進めるための手法が、RFC 6264[†14]として提案されています。RFC 6264で提案されているのは図24.10に示すような、IPv4のみで接続された既存利用者がCGNを経由せずにIPv4インターネットへとつながる環境です。



▶ 図24.10 徐々にCGN化していくアプローチ

IPv4とIPv6のデュアルスタック環境が提供されるユーザ宅のCPE（Customer Premises Equipment）は、IPv4パケットを受け取るとCGNへと転送します。CPEは、IPv4トンネルを利用してIPv6パケットをCGNへと送ります。CGNは、CPEからのIPv4トンネルを経由して送信されてきたIPv6パケットをトンネルから取り出し、IPv6インターネットへと転送します。

この提案のポイントは、ISPの内部ネットワークがすべてIPv4で運用されている点です。CGNを導入できるという意味では、DS-Liteも同様ですが、DS-LiteはISPの内部ネットワークがIPv6シングルスタックで運用されるという違いがあります。

24.6 464XLAT

464XLATは、IPv4をIPv6へと変換し、変換されたIPv6をさらにIPv4へと変換するという2回の変換により、次のような状況に対処する運用手法です。

- IPv6のみで構成され、IPv4インターネットへの接続手法として用意されているのがNAT64のみ

[†14] RFC 6264 : S. Jiang, D. Guo, B. Carpenter, “An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition”, 2011年6月

CC BY NC SA

- IPv4でしか通信できないアプリケーションがあり、何らかの方法でIPv6のみのネットワークを経由してIPv4インターネットと通信する必要がある

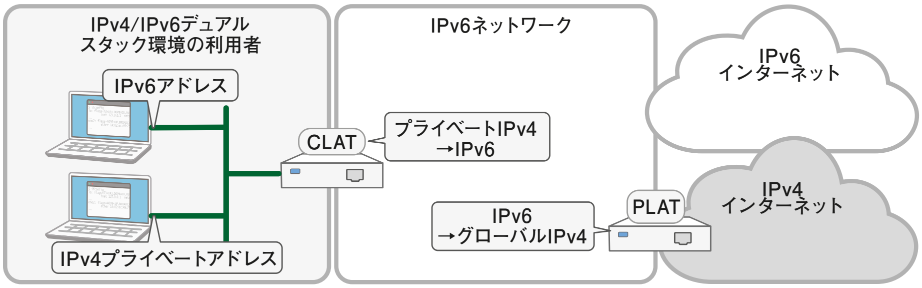
464XLATの仕様はRFC 6877[†15]として発行されています。464XLATのために新しく定義されたプロトコルはなく、既存技術を組み合わせて上記のような状況に対処することを目指しているので、ほぼ既存の製品の組み合わせで実現できるのが特徴です。

464XLATで組み合わされている既存技術は、RFC 6146で定義されたNAT64と、RFC 7915で定義されているSIITです。IPv4上で直接DNSパケットを送受信できる状況が前提なので、RFC 6147で定義されているDNS64は必須ではありません。

なお、464XLATではグローバルIPv4アドレスを持つサーバとの通信が想定されているので、464XLATを介したP2P型通信は実現できません。

24.6.1 464XLATの構成例

464XLATを利用した構成例を図24.11に示します。ユーザ側に設置されるトランスレータをCLAT（Customer side translator）、プロバイダ側に設置されるトランスレータをPLAT（Provider side translator）と呼びます。



▶ 図24.11 464XLAT構成例

CLATに接続されたユーザがIPv6で通信する場合には、CLATからIPv6でそのままやり取りできます。IPv4での通信は、CLATによってIPv6へと変換されたのちにPLATでIPv6からIPv4へと変換されることで、IPv4インターネットとやり取りできます。

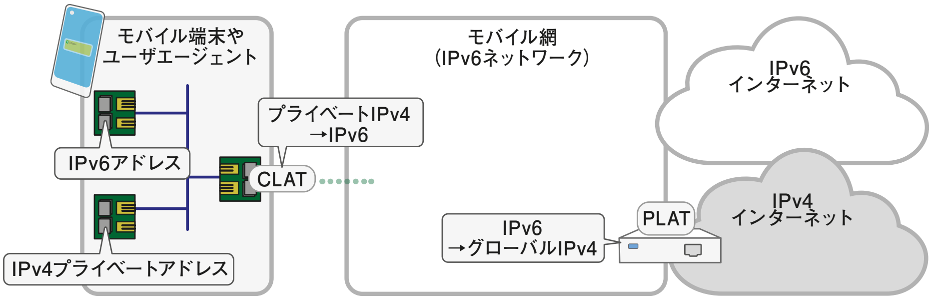
464XLATでは、図24.12のように、3GPPネットワークでの構成例も考慮されています。この場合はCLATがモバイル機器に内蔵されます。これにより、モバイル網そのものはIPv6のみで構成しつつ、IPv4にしか対応していない古いアプリケーションでも通信できるようになります。

24.6.2 ステートフルとステートレスの組み合わせ

プロバイダ側に設置されるPLATはステートフルで、ユーザ側CPEとして設置されるCLATはステートレスです。CLATがステートレスとなっているのは、CPEでの実装を可能な限り軽くするためのようです。

[†15] RFC 6877：M. Mawatari, M. Kawashima, C. Byrne, “464XLAT: Combination of Stateful and Stateless Translation”, 2013年4月

CC BY NC SA



▶ 図24.12　464XLAT 3GPPネットワークでの構成例

PLAT側はステートフルなので、実際の通信が開始されてからポート番号が割り当てられます。そのため、あらかじめ利用するポート番号の範囲が決まっているA+PやMAPなどの他の方式と比べて、IPv4アドレスの使用効率が良いというメリットがあります（A+PやMAPでは、ユーザがまったく使っていなくても特定の範囲のポート番号が専有されてしまいます）。とはいえ、当然のことながら、状態を保持することによる負荷というデメリットもあります。

CC BY NC SA

第25章

プロキシ方式

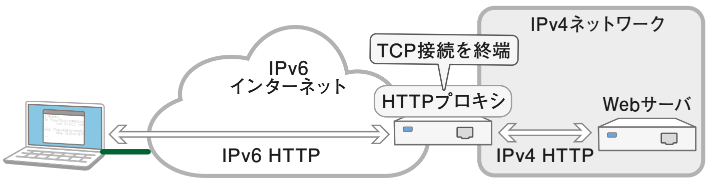
アプリケーションレベルで一度通信を終端したうえでIPパケットを中継することで、IPv4環境とIPv6環境の共存を実現する技術があります。ここでは、それらの手法を**プロキシ方式**としてまとめて扱います。通信内容を理解しつつ必要に応じて調整を行うALG（Application Level Gateway）と呼ばれる仕組みについても本章で扱います。

プロキシ手法の多くはトランスポートプロトコルとしてTCPやUDPを前提にしたステートフルな技術です。各セッションごとに新規ソケットを開いて通信するという実装がよく採用されます。TCPであればIPv6側とIPv4側で個別のソケットを開き、IPv6側とIPv4側でそれぞれ輻輳制御が動作することになります。

プロキシ方式では、一度通信を終端するので、アクセスログに掲載される通信相手が実際の送信元IPアドレスではなく終端を担うプロキシサーバのIPアドレスになります。他の手法にも同様な問題が発生するものはありますが、プロキシ方式の導入でもアクセスログについては十分な考慮が必要になります。

25.1　HTTPプロキシ

IPv6通信を一度終端してIPv4通信を実現するプロキシとして最もよく利用されているのは、おそらくHTTPプロキシでしょう。既存のWebサーバをIPv4で運用しつつ、IPv6によるHTTPプロキシを利用することで、既存システムに手を加えずにWebサーバをIPv6化するサービスもあります。



▶ 図25.1　HTTPプロキシ

CC BY NC SA

HTTPプロキシそのものは、ロードバランサなどによる負荷分散を実現するためによく利用されており、技術そのものは十分に枯れています。グローバルIPv4アドレス空間のHTTPセッションを終端して、負荷分散用に用意されたプライベートIPv4アドレス空間に設置された複数のWebサーバへとコネクションを分散するリバースプロキシとしても広く活用されています。

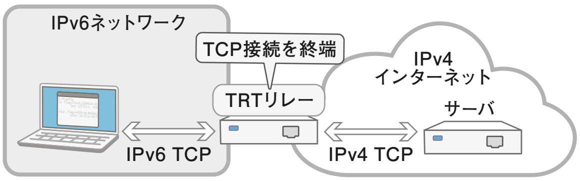
リバースプロキシやロードバランサによって終端されたHTTPセッションの送信元IPアドレスを伝える手段として、`X-Forwarded-For`というHTTPヘッダが利用される場合があります。IPv4からIPv6へのHTTPプロキシや、IPv6からIPv4へのHTTPプロキシでも、この`X-Forwarded-For`ヘッダを利用した送信元IPアドレス伝達が行われます。

2014年に、`X-Forwarded-For`に代わる新しいHTTP拡張として、`Forwarded`ヘッダがRFC 7239[†1]として標準化されました。RFC 7239の`Forwarded`ヘッダは、`X-Forwarded-For`のように宛先だけを記述するのではなく、`by`や`for`、`host`、`proto`などの属性を記述できる仕様になっています。たとえば、`for`であれば次のように指定します。

```
Forwarded: For="[2001:db8:cafe::17]:4711"
```

25.2　TRT方式

IPv6を利用したTCPやUDPセッションを一度終端し、IPv4インターネットに対する通信を仲介する手法のひとつとして、**TRT**（Transport Relay Translator）と呼ばれるものがあります。TRTはRFC 3142[†2]で規定されています。IPv4のみで提供されているサービスに対してIPv6からも接続できるようにすることがTRTの狙いです。RFC 3142は2001年に発行されたInformationalなRFCであり、同じ狙いで開発されたNAT64およびDNS64による方法と比較すると考慮すべき点も少なくありませんが、さまざまな議論の土台となったRFCであるともいえるでしょう。



▶図25.2　TRT方式

TRT方式では、IPv6ユーザが「IPv4アドレスを導出できるIPv6アドレス」でTRTリレーへと接続し、TRTリレーはIPv6での通信を終端しつつIPv4での通信を仲介します。IPv6ユーザが宛先とするIPv6アドレスをどのように決めるかについて、InformationalなRFCであるRFC 3142では具体的な標準フォーマットを規定していません。

†1　RFC 7239：A. Petersson,M. Nilsson, “Forwarded HTTP Extension”, 2014年6月
†2　RFC 3142：J. Hagino, K. Yamamoto, “An IPv6-to-IPv4 Transport Relay Translator”, 2001年6月

CC BY NC SA

25.3 SIP用IPv6プロキシ

ALG機能を含むプロキシとして、SIP用のIPv6変換用プロキシがRFC 6157[†3]で定義されています。SIPはVoIPなどで必要となる呼制御プロトコルなので、通話相手のIPアドレス情報などをペイロードとしてプロトコル中に抱えています。そのため、NAT64およびDNS64によるIPv4/IPv6変換や、TRTによる単純なIPv4/IPv6変換だけでは、SIPセッションのIPv4/IPv6変換ができません。そこで、プロトコル内部を解釈したうえでのIPv4/IPv6プロキシとして、RFC 6157が規定されました。

---

†3 RFC 6157：G. Camarillo, K. El Malki, V. Gurbani, “IPv6 Transition in the Session Initiation Protocol (SIP)”, 2011年4月

CC BY NC SA

[書籍版では空白のページです]

付録A

NTT NGNでのIPv6

NTT東日本とNTT西日本（NTT東西）では、フレッツ（FLET'S）と呼ばれるデータ通信サービスを提供しています。このサービスの基盤となるネットワークは、**NTTフレッツ網**や**NTT NGN**などと呼ばれています。このネットワークは、インターネットとは切り離されている「閉域網」です。日本電信電話株式会社等に関する法律（通称NTT法）による制限で、NTT東西ではインターネット接続サービスを利用者に対して直接提供することが許されていないからです。

閉域網ではありますが、NTT NGN内ではIPv6グローバルユニキャストアドレスが利用されています。東西NTTが展開するフレッツ光の回線を契約すると、この閉域網からIPv6アドレスが割り振られる仕組みです。このIPv6アドレスで、NTT NGN内で閉じたサービスではなくIPv6インターネットへの接続を行うには、ISPとの契約が必要になります。

NTT NGNにおけるIPv6の扱いは非常に特殊であり、そのIPv6をめぐる特殊な事情は、海外で「日本のIPv6問題」と表現されることもありました。「日本のIPv6問題」は、厳密に言うと2種類に分けて考えられます。NTT閉域網IPv6フォールバック問題と、NTT NGN IPv6マルチプレフィックス問題です。両方とも原因は同一であり、互いに密接に絡み合っている問題ですが、現象や周辺状況が異なっているので、本書では2種類の別々の問題として扱います。

現在のNTT NGNでは、これらの問題を解決するために、IPv6 IPoEとIPv6 PPPoEという2つの異なる手法が提供されています。以降の解説では、これらの手法が必要となる技術的な要因を説明するために、「NTT閉域網IPv6フォールバック問題」および「NTT NGN IPv6マルチプレフィックス問題」という表現をしています。繰り返しになりますが、これらの「問題」はIPv6 IPoEとIPv6 PPPoEを採用することで解決されています。

この付録では、NTT NGN経由でIPv6インターネット接続を実現するうえで何が問題であったかと、その解決策としてのIPv6 IPoEおよびIPv6 PPPoEについて説明します。

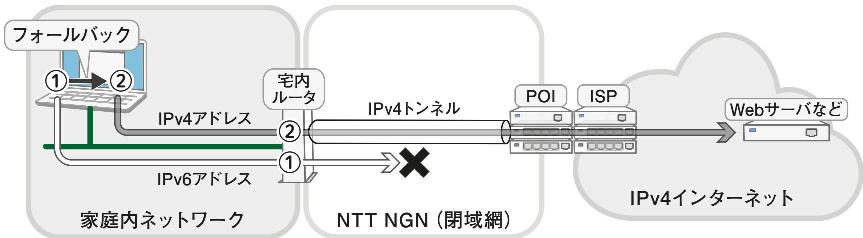
A.1 NTT閉域網IPv6フォールバック問題

NTT NGN上で提供されているBフレッツ、フレッツ光ネクスト、光プレミアム（NTT西日本）の各サービスでは、ユーザセグメントにIPv6アドレスが配布されます。しかし、そのIPv6

cc BY NC SA

アドレスではIPv6インターネットと通信できません。そのIPv6アドレスは、割り振りとしてはグローバルユニキャストアドレスですが、NTT NGNがIPv6インターネットと接続されていないからです。

これらのNTTフレッツ網で提供されているサービス上で、ISPを経由してIPv4インターネットに接続しているとします。このとき、IPv6対応のPCを使用していてIPv6が有効になっている場合、インターネットに接続する際に「IPv6からIPv4へのフォールバック」が必ず発生するという問題がありました。これは、NTT NGNからインターネットと通信できないグローバルユニキャストアドレスが割り振られていることで、宛先アドレスとしてIPv4アドレスよりIPv6アドレスが優先されてしまう可能性が高かったからです（13.1節で説明した旧RFC 3484におけるデフォルトIPv6アドレス選択のルールを参照）。ISPでIPv6による接続サービスを申し込んでいない場合、つまりユーザのPCにIPv6でのインターネット接続性がない状態でNTT NGNによるIPv6アドレスが設定されている状況を図A.1に示します。



▶ 図A.1 NTT閉域網IPv6フォールバック

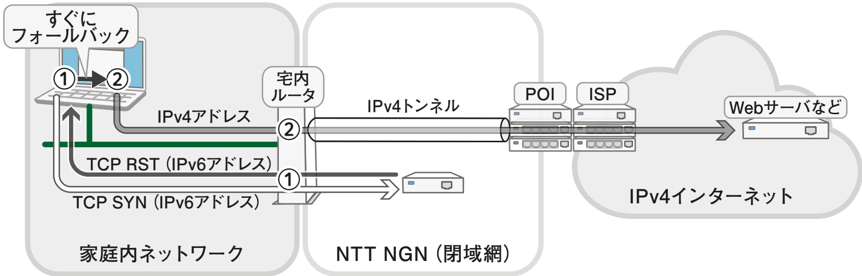
この問題は、優先的に使われる設定になっているグローバルユニキャストアドレスで運用されているネットワーク（NTT NGN）が、IPv6インターネットに接続されていないために発生していたものです。このIPv6からIPv4へのフォールバックを回避するためのアプローチとして最も単純なのは、常にIPv4のみが利用されるようにIPv6を利用しない設定にする方法でしょう。しかし、それではNTT NGN内においてIPv6を利用したサービスが使えなくなります。そこで現在では、IPv6 IPoEおよびIPv6 PPPoEという、インターネットに接続されたIPv6を利用する方法が採用されています。

IPv6 IPoEおよびIPv6 PPPoEについてはA.3節で説明します。ここではNTT閉域網IPv6フォールバック問題への短期的な対策や背景について補足します。

A.1.1 TCP RST

NTT NGN閉域網では、IPv6からIPv4へのフォールバックにかかる時間を短縮するために、NTT NGN閉域網の外にあるIPv6アドレス宛のTCP通信に対してはTCP RSTパケットを送付するという手法が考案されたことがあります。TCP RSTパケットを送付することで、ユーザが開始しようとしたIPv6でのTCP接続の確立が早期に失敗し、IPv4へとフォールバックするまでの時間を短縮できます。

ただし、この対策はあくまでもTCPを利用した接続に対してしか利用できません。また、TCP RSTパケットがネットワークで喪失する可能性があること、IPv6でTCP RSTを受け取っ



▶ 図A.2　NTT閉域網IPv6 TCP RST

て「接続失敗」となった場合にIPv4でTCP接続を再度試みるかどうかはアプリケーションの実装依存であることといった点にも注意が必要です。

A.1.2　NTT閉域網IPv6フォールバック問題は新しい問題ではない

NTT東日本がBフレッツ回線にIPv6アドレスを割り当て始めたのは2007年2月のことです。2007年時点で、NTT東日本のWebサイトには下記のような注意事項が記載されていました。

> インターネット上には、IPv6とIPv4の両方で公開されているホームページが存在しており、このようなホームページへアクセスする場合には、初回表示に数十秒かかる場合がございます
> ...（中略）...
> 【対処方法2】
> IPv6をはずすことで回避可能です。設定手順は、以下のとおりです。

BフレッツやNTT西日本のフレッツ光プレミアムだけでなく、2008年に開始されたNTT NGNサービスであるフレッツ光ネクストでも、NTT NGN閉域網のIPv6アドレスが割り当てられています。

NTT閉域網IPv6フォールバック問題が大きく取り上げられるようになったのは2008年前後のことです。その背景には、この時期にIPv4アドレスの在庫枯渇が現実のものとなり、IPv6に関する取り組みが増えたことがあります。それまでは、NTT閉域網IPv6フォールバック問題が発生し得る環境であったとしても、インターネット上にあるサーバのほうがIPv6に対応していなかったので、ユーザ側がIPv6による接続を試みることがなかったのです。世界中で多くのサーバがIPv6対応を実施し、権威DNSサーバにAAAAが登録されたことで、かなり前から存在していた問題が顕在化したと考えられます。

実際、NTT閉域網IPv6フォールバック問題が世界的に有名になった背景には、World IPv6 Dayにおいて多くのコンテンツ事業者が期間限定でIPv6によるサービスを提供した際、多数のユーザ側がIPv6での接続を試みるような状況が発生し、AAAAレコードを登録したWebサーバ側でも通信確立までの遅延が観測されたという事件がありました。

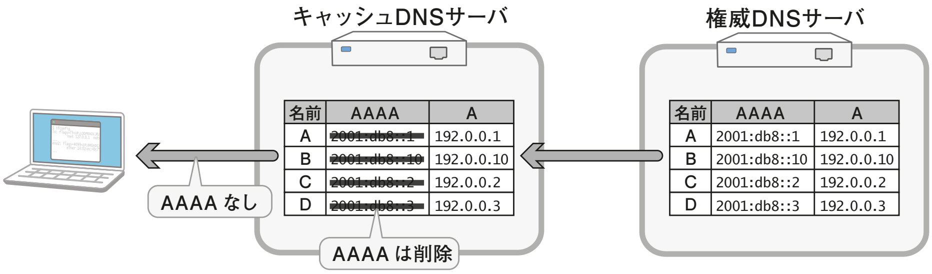
cc BY NC SA

A.1.3　クライアントサイドでのHappy Eyeballsによる対処

NTT閉域網IPv6フォールバック問題では、IPv6によるインターネットへのアクセスが必ず失敗します。しかし、そもそもこのようなフォールバックはNTT NGN閉域網に固有の問題ではありません。IPv4とIPv6のデュアルスタック環境であればどこでも発生し得る問題です。たとえば、IPv4インターネットでは発生していないような大規模な障害がIPv6インターネットの経路上だけで発生したような状況でも、類似の問題が発生すると考えられます。そのような状況を避けるため、Happy Eyeballsという仕様が提案されています（15.2節参照）。

A.1.4　AAAAフィルタリング

AAAAフィルタリングは、ISPにあるキャッシュDNSサーバで、DNS応答に含まれるAAAAレコードを抜いてしまうという手法です。キャッシュDNSサーバがAAAAレコードを除外するので、結果的にユーザがIPv6での通信を試みなくなります。「ユーザにIPv6を使わせない」方法だといえます。



▶ 図A.3　AAAAフィルタリング

AAAAフィルタリングは、2011年のWorld IPv6 Day前後に試験的に実施されました。実施したのは日本国内のISP事業者やIX事業者です。World IPv6 Day中のNTT閉域網IPv6フォールバック問題の発生がかなり抑えられました。

AAAAフィルタリングはIPv6からIPv4へのフォールバック発生に対する解決策ではなく、短期的な暫定的対応であるという点に注意が必要です。IPv6対応のための時間を稼ぐための手法でしかありません。

2012年のWorld IPv6 Launch前後には、ISPのキャッシュDNSサーバにおけるAAAAフィルタと同時に、コンテンツ事業者におけるDNS IPv6ブラックリスティングも実施されました。権威DNSでのブラックリスト方式については17.5.2項を参照してください。

A.2　NTT NGN IPv6マルチプレフィックス問題

「NTT NGN IPv6マルチプレフィックス問題」は、NTT NGNからIPv6インターネットに接続されていないグローバルユニキャストアドレスがユーザ端末に設定された状態で、その端末にISPからもIPv6アドレスが設定されてしまうことで発生していた問題です。問題の解決をめぐっては全部で4つの案が検討され、そのうち案2および案4と題されていたものが、それぞれIPv6 PPPoEおよびIPv6 IPoEとして、2011年に一般向けサービスが開始されています。本

CC BY NC SA

書執筆時点ではすでに解決策が広く普及しており、その意味では「問題」ではありません。

NTT NGN IPv6 マルチプレフィックス問題は、「ISP がどのように IPv6 サービスを提供すべきであるかに関する議論」だったといえます。2008 年頃には、この問題を契機に、ISP がどのようにして NTT NGN で IPv6 サービスを提供すべきかについての議論が活発に交わされました。ここでは、現状の解決策である、IPv6 PPPoE および IPv6 IPoE について説明する前に、議論が交わされていた当時の「NTT NGN IPv6 マルチプレフィックス問題」が具体的にどういった課題だったかをまとめます。

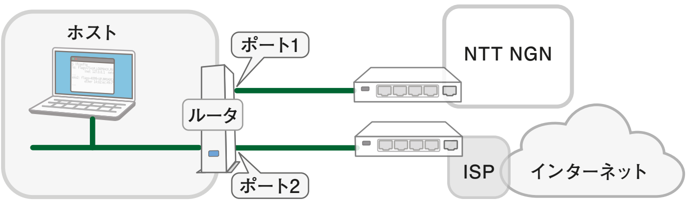
A.2.1　問題が発生する状況

NTT NGN IPv6 マルチプレフィックス問題は、IPv6 IPoE や IPv6 PPPoE といった解決策を使わない場合に NTT NGN で発生してしまうことが想定される、日本のネットワーク固有の問題です。それが発生する背景には、NTT 法による制限、NTT 東西のネットワーク設計および判断に加えて、IPv6 がそもそも抱えている課題があります。

NOTE

RFC 7157[†1] では、IPv6 におけるマルチプレフィックス問題が一般的に議論されています。このRFC は、NTT NGN における IPv6 マルチプレフィックス問題のことを IPv6 が本質的に抱えている課題の表面化であると捉えた NTT 社員が中心になって書かれたものです。

IPv6 IPoE や IPv6 PPPoE といった解決策がない状態で、NTT NGN に接続しているユーザに対して ISP が IPv6 インターネット接続サービスを提供してしまうと、図 A.4 のような状態になると考えられます。この環境では、NTT NGN による IPv6 プレフィックスと ISP による IPv6 プレフィックスの両方がホストに対して設定される可能性があります。



▶ 図 A.4　NTT NGN IPv6 マルチプレフィックス問題が発生する状況

この環境は、13.2.1 項で紹介したシナリオ 2 に類似した環境です（図 13.2 と比べてみてください）。そのため、そこで紹介されているのと同じ課題を抱えています。そのうえで NTT NGN に固有の話として、IPv6 インターネットと接続していないことから、NTT NGN IPv6 ネットワークを利用することを選択したうえで IPv6 インターネットと通信を行おうとすると通信が必ず失敗してしまうという問題が発生します。このままでは NTT NGN における IPv6 インターネット接続サービスを ISP が一般利用者向けに提供できないということで、解決策が議論され

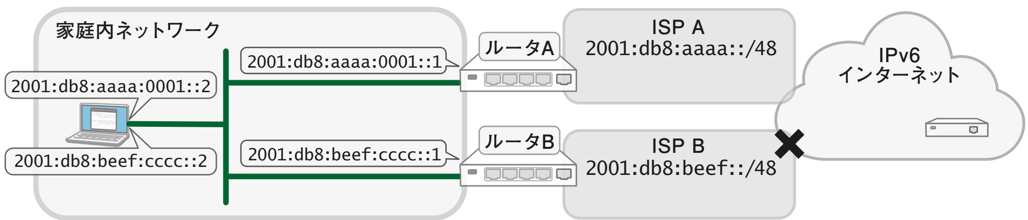
†1　RFC 7157：O. Troan, D. Miles, S. Matsushima, T. Okimoto, D. Wing, “IPv6 Multihoming without Network Address Translation”, 2014 年 3 月

ました。解決策が決まるまではNTT NGN経由のIPv6インターネット接続サービスが実施できないこと、どのような解決策を採用するかによって日本におけるISPの事業形態が大きな影響を受けることが予想されることから、この「問題」をめぐっては活発に議論が交わされました。最終的にはIPv6 IPoEとIPv6 PPPoEという解決策が決定し、現在ではすでに問題は解消しています。

A.2.2　通信ができないIPv6環境での例

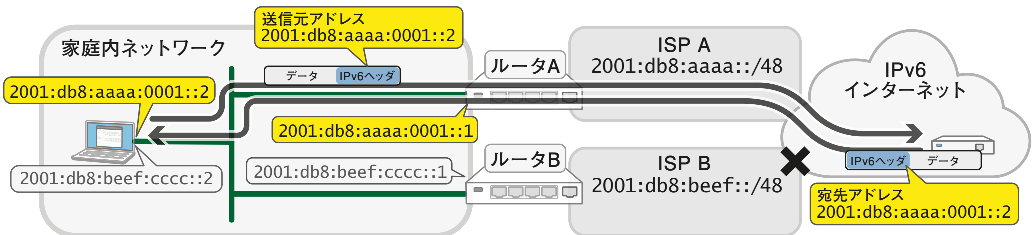
IPv6では1つのネットワークインターフェースに複数のネットワークプレフィックスが設定可能ですが、そのようなマルチプレフィックス状態であっても、各通信では単一のIPv6アドレスが利用されます。このとき、IPv6マルチプレフィックス状態のホストは、何らかの方法で通信に利用するIPv6アドレスを選択しなければなりません。その選択によって通信結果が異なる場合もあります。

場合によってはIPv6アドレス選択によって悪影響が生じることもあります。図A.5では、ISP AとISP Bの両方につながった家庭内にあるパソコンに、ISP AからのIPv6アドレスとして2001:db8:aaaa:0001::2が、ISP BからのIPv6アドレスとして2001:db8:beef:cccc::2が両方とも設定されています。このとき、何らかの理由で、ISP BにおけるIPv6インターネットへの接続性に一時的な障害が発生したとします（IETFでの議論やRFCにおいて「“broken” IPv6」と表現される状況です）。



▶ 図A.5　IPv6アドレス選択が悪影響を与える例1

このような環境下で、家庭内にあるパソコンが送信元IPv6アドレスとして2001:db8:aaaa:0001::2を選択し、ルータA経由でIPv6インターネット上にあるサーバと通信する場合には、通信は正常に行われます。

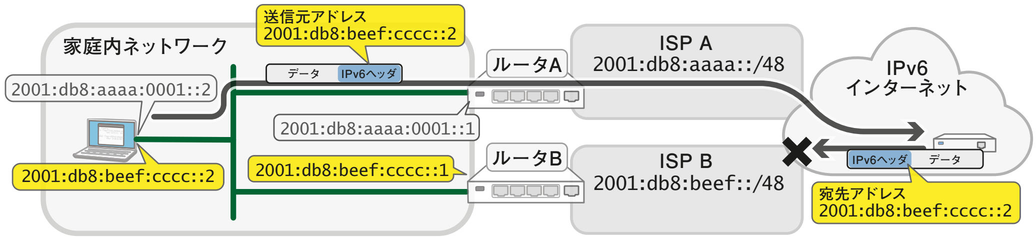


▶ 図A.6　IPv6アドレス選択が悪影響を与える例2

しかし、家庭内にあるパソコンが送信元IPv6アドレスとして2001:db8:beef:cccc::2を選択し、ルータA経由でIPv6インターネット上にあるサーバとの通信を試みる場合には、家庭

CC BY NC SA

内のパソコンからサーバまでのパケットは到達するものの、サーバから家庭までのパケットは到達せず、通信が成立しません。



▶図 A.7　IPv6 アドレス選択が悪影響を与える例 3

なお、この環境下では、家庭内のパソコンがルータ B を経由して通信しようとすると、送信元 IPv6 アドレスの選択にかかわらず、すべての通信が失敗してしまいます。図 A.7 の状況で家庭内のパソコンが通信を成功させるには、RFC 7157 によると以下の 3 点が重要です。

- 適切な次ホップを選択してパケットを送信できること
- 適切な送信元 IPv6 アドレスを利用してパケットを送信できること
- 適切なキャッシュ DNS サーバを選択できること

送信元 IPv6 アドレスの選択と経路の選択が互いに独立している点がポイントだといえるでしょう。

A.3　IPv6 PPPoE と IPv6 IPoE

NTT NGN IPv6 マルチプレフィックス問題に対しては、解決策として、IPv6 PPPoE と IPv6 IPoE の 2 種類が利用されています。IPv6 PPPoE か IPv6 IPoE を利用することで、NTT NGN からの IPv6 アドレスが利用者のネットワーク内には割り当てされなくなるので、複数のグローバルユニキャストアドレスが設定される状態が避けられます。これにより、NTT NGN IPv6 マルチプレフィックス問題を発生させていた要因が解消することになります。ユーザが利用するホストに対して複数のプレフィックスを提供せずにシングルプレフィックスにすることで問題を回避する方法だといえるでしょう。

NTT 東西で提供されている IPv6 PPPoE と IPv6 IPoE の違いを表 A.1 にまとめます。なお、表中の VNE は Virtual Network Enabler のことで、NTT 東西と契約して他の ISP に代わって IPv6 サービスを提供する事業者を指します。

cc BY NC SA

▶ 表A.1　NTT東西で提供されているIPv6 PPPoEとIPv6 IPoE

項目	IPv6 PPPoE	IPv6 IPoE
旧称	トンネル方式（案2）	ネイティブ方式（案4）
ユーザの宅内のIPv6アドレス	ISPのIPv6アドレス	VNEからNTTに預けられたIPv6アドレス
ユーザの宅内で必要となる機器	IPv6トンネルアダプタ	特になし
ユーザのトラフィックの経路	ISP経由	VNE経由
PPPoE	IPv4とIPv6で別々のPPPoEセッション	IPv6用のPPPoEは不要
特徴	トンネルアダプタ内でNAT66を実施	NGN内で送信元IPv6アドレスをもとにしたPolicy Basedルーティングを実施

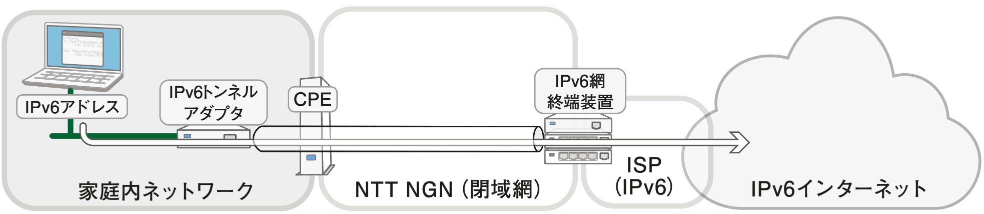
NTTフレッツネクストを利用するユーザに対してISPがIPv6サービスを提供するときに、どちらの方式を利用するのかは、各ISPごとの判断となります。ユーザは、利用しているISPの対応状況を見ながら、IPv6 PPPoEとIPv6 IPoEのどちらを利用するのかを状況に応じて判断し、申し込みを行う必要があります。

なお、技術的には、各ユーザがIPv4用のISPとIPv6用のISPを別々に利用することも可能です。

A.3.1　IPv6 PPPoE

IPv6 PPPoEは、NTTフレッツ網でIPv4インターネット接続サービスと同様にPPPoEを利用する方式です。「トンネル方式」という名前のとおり、IPv6 PPPoEでは、図A.8のように、ユーザの宅内にあるIPv6トンネルアダプタからNTT NGN内にあるIPv6終端装置までトンネルが張られます。IPv6 PPPoEは、IPv4におけるインターネット接続サービスの多くと同様に、ISPとの接続や認証にPPP（Point-to-Point Protocol）を利用します。

IPv6終端装置はISPのネットワークと接続されており、ISPを経由してIPv6インターネットとの通信が可能になります。

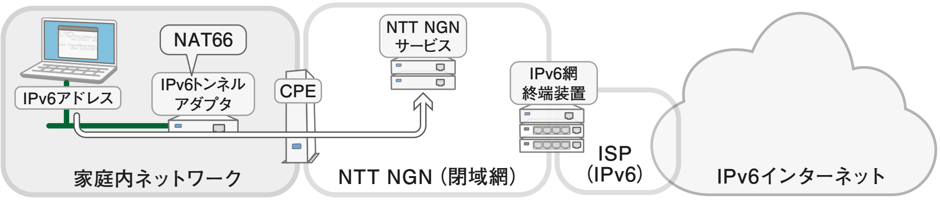


▶ 図A.8　IPv6 PPPoE

IPv6トンネルアダプタに接続しているユーザの宅内のLANに対しては、ISPからIPv6のネットワークプレフィックスが割り当てられます。このときユーザの宅内で利用されるIPv6のネットワークプレフィックスは、ISPから割り当てられるもののみです。

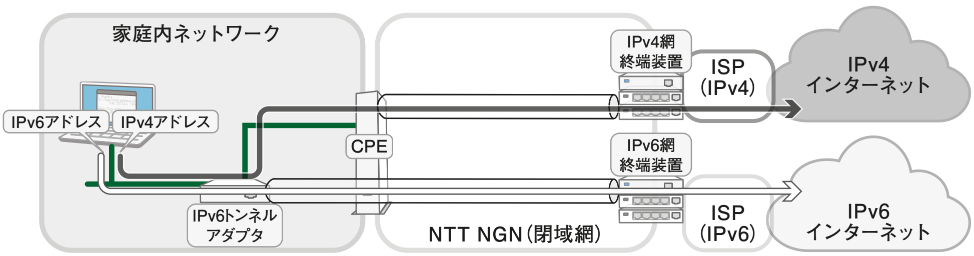
IPv6トンネルアダプタは、IPv6インターネット宛のトラフィックをトンネル経由でISPへ

と転送します。NTT NGN内を宛先とするトラフィックについてはトンネルを経由させず、NAT66を利用することで送信元IPv6アドレスをNTT NGNにおけるIPv6のネットワークプレフィックスへと変換してから転送します。IPv6トンネルアダプタによって、NTT NGN宛の通信とIPv6インターネット宛の通信が両方とも可能になります。



▶ 図A.9 IPv6 PPPoEで行われるNAT66

一般のユーザは、IPv6インターネットだけではなくIPv4インターネットもよく利用します。その場合には、IPv6 PPPoEに加えて、IPv4 PPPoEも利用することになります。具体的には、図A.10のように、IPv4 PPPoEの接続はホームゲートウェイ（CPE）が担当し、IPv6 PPPoEはIPv6トンネルアダプタが担当するという構成になります。IPv4 PPPoEとIPv6トンネルアダプタの両方の機能が搭載されたホームゲートウェイが使われる場合もあります。



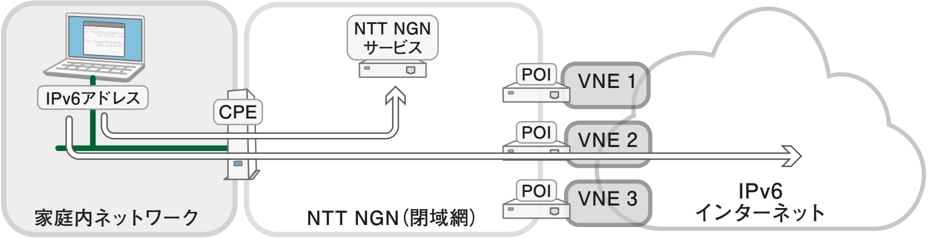
▶ 図A.10 IPv4 PPPoEとIPv6 PPPoE

A.3.2 IPv6 IPoE

IPv6 IPoEでは、VNE（Virtual Network Enabler）と呼ばれる事業者がNTT東西と契約し、他のISPに代わってIPv6サービスを提供します。VNEは、IPv6ネットワークを運用しつつNTT NGN網との相互接続を行います。IPv6 IPoEでは、VNEがISPに対してローミングサービスを提供するわけです。IPv6 IPoEを申し込んだユーザには、VNEからNTTに預けられたIPv6アドレスが割り当てられます。ISPは、VNEと契約してVNEの顧客となることで、ユーザに対してIPv6サービスを提供します。IPv6 IPoEにおけるISPの役割は、IPv6ネットワークを直接運用することではなく、アカウント管理や課金などです。

IPv6 IPoEは、図A.11のような構成で運用されています。

IPv6 IPoEを利用するユーザからのIPv6トラフィックは、NTT NGN内で送信元IPv6アドレスをもとにしたPolicy Basedルーティングされることにより、VNEとの相互接続地点に設定さ

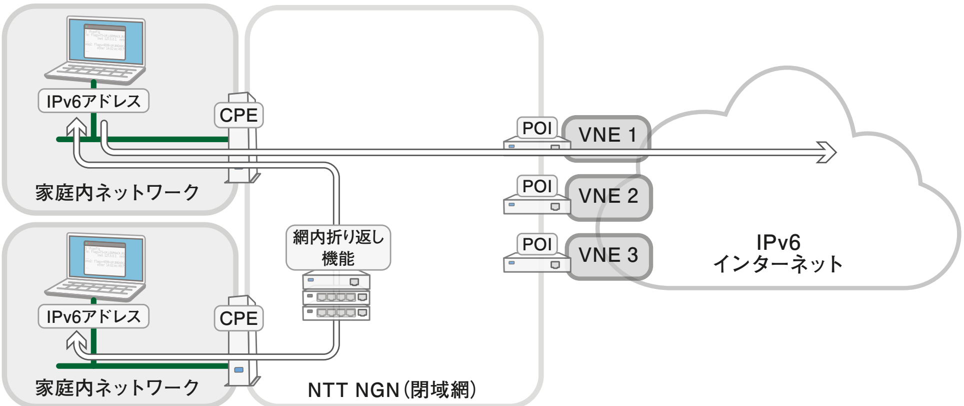


▶ 図A.11　IPv6 IPoE

れたゲートウェイルータへと転送されます。「送信元IPv6アドレスをもとにしたPolicy Basedルーティング」というのは、送信元アドレスに応じて転送先が決定されていくことを意味しています。通常、IPv6ユニキャストパケットは、IPv6ヘッダに記載された宛先IPv6アドレスに応じて転送されますが、IPv6 IPoEでは、送信元IPv6アドレスも考慮されたルーティングが行われるのです。このような仕組みになっているのは、どのVNEを経由してIPv6インターネットへと転送するかを、各ユーザのIPv6アドレスに応じて決定するからです。相互接続地点のゲートウェイルータでは、各パケットの送信元アドレスを見て、適切なVNEへと転送します。

IPv6インターネットとの通信でなく、NTT NGNサービスとのIPv6による通信では、ユーザからのIPv6パケットの宛先IPv6アドレスがNTT NGNサービスのIPv6アドレスになっています。そのようなIPv6パケットは、NTT NGN内のサーバへとルーティングされます。

さらに、IPv6 IPoEによる直接的な機能ではありませんが、IPv6 IPoEではNTT NGNにおける網内折り返し機能も利用できます[†2]。これは、IPv6 IPoEを利用する機器同士の通信において、VNEを経由せずにNTT NGN内で通信が可能になるというものです。ただし、NTT東西をまたがるような通信に関しては、VNEを経由しての折り返し通信となります。



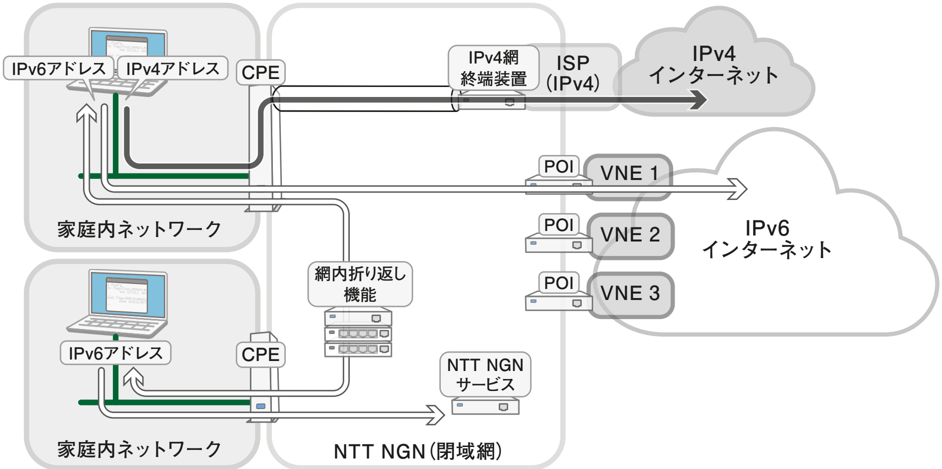
▶ 図A.12　IPv6 IPoE（網内折り返し通信）

IPv6 IPoEを利用する場合のIPv4インターネット接続サービスとIPv6関連サービスをまとめると図A.13のようになります。各家庭内にあるCPE（ホームゲートウェイとも呼ばれます）

†2　網内折り返しは「フレッツ・v6オプション」に契約する必要があります。ISPの契約とは独立しています。

CC BY NC SA

を経由して、IPv4サービスとIPv6サービスの両方が提供されます[†3]。IPv4サービスのためのIPv4パケットは、ホームゲートウェイからIPv4網終端装置までNTT NGN内をトンネルで転送され、そこからISPまで運ばれます。



▶ 図A.13 IPv6 IPoE（全体像）

IPv6 IPoEは、IPv6 PPPoEのIPv6トンネルアダプタのような追加機器を必要とせず、NTT東西側の設定によって実現します。IPv6 IPoEを利用するユーザに対しては、VNEからIPv6アドレスが割り当てられるので、NAT66が必要なIPv6 PPPoEと比べると仕組みも比較的シンプルです。

ただし、技術的制約から、VNEの総数は制限されています。IPv6 IPoEのサービス開始時点では、BBIX株式会社、インターネットマルチフィード株式会社、日本インターネットエクスチェンジ株式会社（第1刷発行当時は日本ネットワークイネイブラー株式会社）の3社がVNEとして事業を展開していました。BBIXはソフトバンク、日本ネットワークイネイブラー株式会社（第1刷発行当時）はKDDI、インターネットマルチフィードはNTTの系列会社であり、日本国内の大手3キャリアで3つの枠を分け合っていた格好です。その後、最大16社までVNEを増やせるような方向で検討が進められ[†4]、2021年4月21日時点では9社がVNEとして承諾を受けています[†5]。

- 2009年12月4日（公表段階）[†6]
  - BBIX株式会社
  - インターネットマルチフィード株式会社
  - 日本インターネットエクスチェンジ株式会社（第1刷発行当時は日本ネットワークイネイブラー株式会社）

---

†3 CPEがなくても、IPv4サービスとIPv6サービスの両方が提供されます。
†4 https://www.soumu.go.jp/main_content/000033625.pdf
†5 https://www.ntt-east.co.jp/info-st/ipoe_menu/index.html
†6 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6sentei091204.pdf

CC BY NC SA

- 2016年5月28日[†7]
  - ビッグローブ株式会社
  - 株式会社朝日ネット
- 2016年9月29日[†8]
  - エヌ・ティ・ティ・コミュニケーションズ株式会社
- 2017年7月6日[†9]
  - フリービット株式会社（現在は株式会社ファミリーネット・ジャパン）
- 2018年3月5日[†10]
  - アルテリア・ネットワークス株式会社
- 2020年7月17日[†11]
  - 楽天モバイル株式会社

A.3.3　BフレッツでのIPv6インターネット接続サービス

Bフレッツとフレッツ光ネクストには違いがあります[†12]。さらに、NTT西日本にはフレッツ光プレミアムという別網もあります。NGN（Next Generation Network、次世代ネットワーク）と呼ばれることもあるのは、フレッツ光ネクストです[†13]。

NGNは、保守コストが上昇している交換機ではなく、汎用的なIP製品を活用して電話網を構築し直したものです。NGNでは、IP製品を使ってユニバーサルサービスを提供するので、電話以外のサービスも同時に行えるという特徴もあります。

IPv6 PPPoEとIPv6 IPoEが提供されているのは、フレッツ光ネクスト（NGN）です[†14]。一方で、Bフレッツ、フレッツ光プレミアム、フレッツ光ネクストのすべてで、閉域網からIPv6アドレスが提供されています。そのため、フレッツ光ネクスト以外では、公式な手法としてIPv6インターネットに接続する方法は提供されていませんでした。Bフレッツからフレッツ光ネクストへのサービス移行によってIPv6インターネット接続サービスが提供されたことになりま

†7 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6shoudaku20160510.pdf
†8 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6shoudaku20160929.pdf
†9 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6shoudaku20170706.pdf
†10 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6shoudaku20180305.pdf
†11 https://www.ntt-east.co.jp/info-st/mutial/ngn/IPv6shoudaku20190717.pdf
†12 Bフレッツは2021年1月に終了しています。https://www.ntt-east.co.jp/release/detail/20180409_01.html
†13 本書とは関係ありませんが、総務省などで主に光スイッチングなどの研究でNwGN（新世代ネットワーク）という単語が出てきますが、それはNGNとは別の話です。次世代と新世代で、非常にややこしいのですが、NTTフレッツ網でのIPv6が関係するのは「次世代」のほうです。
†14 BフレッツとフレッツADSLでIPv6を利用できる「FLET'S.Net（フレッツ・ドットネット）」というサービスもありました。https://www.ntt-east.co.jp/release/0312/031218c.html

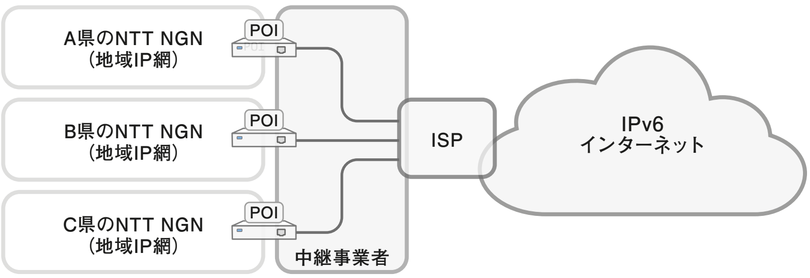
CC BY NC SA

す[†15]。

A.3.4 IPv6 PPPoE と IPv6 IPoE のビジネス構造的な違い

IPv6 PPPoE と IPv6 IPoE の間には、通信そのものの技術的な違いだけではなく、ビジネス構造による違いもあります。NTT NGN IPv6 マルチプレフィックス問題に対して4つの解決案が提案され、そのうち案2（IPv6 PPPoE）と案4（IPv6 IPoE）が採用された背景には、このビジネス構造の違いという側面がありました。

ISP が、NTT 東西の保有する NGN を利用して IPv4 インターネット接続サービスを提供する場合、ISP は図 A.14 のように各県にある POI（Point Of Interface）で各家庭に対して IPv4 インターネット接続サービスを提供します[†16]。ISP では、伝送サービスを提供する中継事業者と契約することで、各 POI から自社の IPv4 ネットワークへの接続を実現しています。中継事業者として伝送サービスを提供する事業者としては、NTT グループ内で長距離会社として民営化された NTT コミュニケーションズがあります。



▶ 図 A.14 NGN を利用したインターネット接続サービスの事業形態

NTT 東西が提供する IPv6 PPPoE は、ISP が NGN を利用して提供する IPv4 インターネット接続サービスの形態に近い形態で IPv6 インターネット接続サービスを実現する方法です。IPv6 PPPoE では、ISP が各 POI でユーザ宅との接続を受け取り、自社 IPv6 ネットワークまでの伝送サービスを中継事業者から購入します。POI にて、ユーザに対するサービス提供用の設備を ISP が設置し、ISP が IPv6 通信を実現します。

これに対し、IPv6 IPoE の接続形態は IPv6 PPPoE とはまったく異なります。IPv6 IPoE では、ISP が NGN と接続する箇所が各県の POI ではなく、集約された相互接続地点になります（図 A.15）。ISP の役割はパケットを直接処理することではなく、ユーザのアカウントや課金を管理することに特化されます。

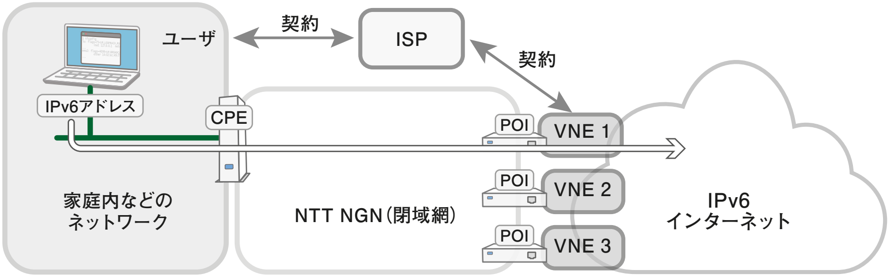
ISP にとって、IPv6 IPoE での接続形態は、IPv4 における ISP 用のローミングサービスに似ています。各県の POI で設備を用意したうえで伝送サービスを購入するには大きな投資が必要なので、中小 ISP はローミング事業者からローミングサービスを購入することで ISP 事業を営みます（図 A.16）。ローミング事業は、「ISP 用の ISP」と表現されることもあります。日本国

---

†15 https://www.ntt-east.co.jp/release/detail/20130725_01.html

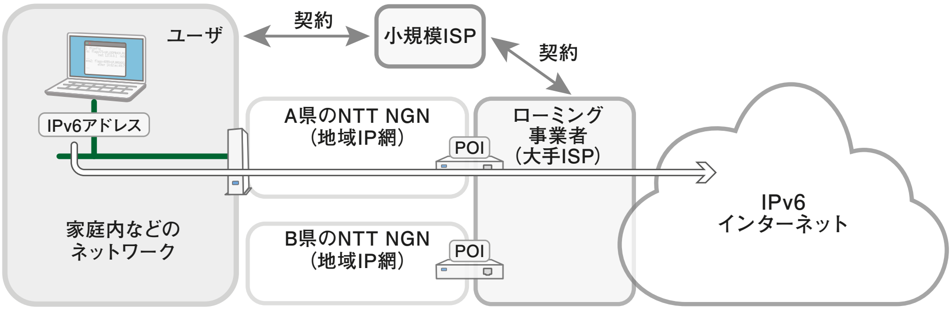
†16 NGN の POI は日本全国に用意されていますが、ISP が日本全国のすべての POI に対応する必要があるわけではありません。

cc BY NC SA



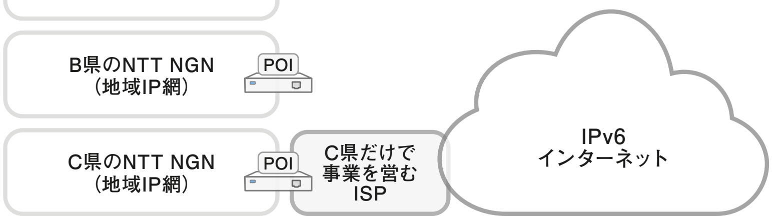
▶ 図 A.15　IPv6 IPoE の事業形態

内でローミング事業を行っているISPとしては、IIJ、FreeBit、アルテリア・ネットワークス（VECTANT）、NTT PCなどが挙げられます。



▶ 図 A.16　ローミングサービスの事業形態

全国規模のISP事業を前提とせず、特定の県内だけで事業を営む小規模ISPでは、図A.17のように必要なPOIだけに設備を設置することによってIPv6 PPPoEによるサービスを提供することも可能です。



▶ 図 A.17　小規模ISPによるIPv6 PPPoEによるサービスの形態

IPv6 PPPoEとIPv6 IPoEが両方とも提供されているのは、このような両方式のビジネスモデルの違いによる面があったと考えられます。

なお、本書執筆現在では、IPv6 PPPoEによる提供よりもIPv6 IPoEによる提供のほうが主流になりつつあります。

NTTの成り立ち

NTTの正式名称は日本電信電話株式会社ですが、その前身は日本電信電話公社という国営公社でした。1985年に日本電信電話公社が民営化され、日本電信電話株式会社が設立されました。

1984年に制定された「日本電信電話株式会社等に関する法律」（通称NTT法）に対しては、これまで何度か改正が行われています。1997年には「日本電信電話株式会社法の一部を改正する法律」が成立し、NTTは、持ち株会社と地域会社2社と長距離会社に分離されました。地域会社2社がNTT東日本とNTT西日本、長距離会社がNTTコミュニケーションズです。

地域会社2社は、同一都道府県の区域内における電気通信業務を経営することが目的とされました。そのためのネットワークが「地域IP網」です。

A.3.5 IPv6 IPoEとIPv6 PPPoEの開始時期の違い

当時、「案2」とされたIPv6 PPPoEは2011年6月1日に開始され、「案4」とされたIPv6 IPoEは2011年7月26日に開始されました。

開始時期が異なっている背景は、NTT NGNにおけるIPv6マルチプレフィックス問題の議論と、それを受けて出された総務省の答申[†17]からうかがい知ることができます。答申書の最初のページには以下のようにあります。

> ＮＴＴ東西に対し、トンネル方式の提供開始時期がネイティブ方式の提供開始時期より遅れることのないように努めることを要請すること。（考え方5）

さらに、この「考え方5」に対応する「意見」が以下のように記述されています。

> 意見5 ネイティブ方式が先に開始されてしまうと、アダプタの追加が必要なトンネル方式に乗り換える可能性は非常に低いため、トンネル方式を先行提供するべき。

ネイティブ方式が有利になってしまうので、アダプタが必要な案2を先に開始できるようにすべきであるという考え方です。当初、NTT NGN IPv6マルチプレフィックス問題に対する解決方法は、案1、案2、案3の3種類でした。しかし、ISP側との折り合いがつかず、落としどころとして、数に制限があるネイティブ方式の案4が登場したという背景があります。そのため、実際に案4を実現するための条件として案2が先にという意向があったのではないかと筆者は推測しています。

NTTは、なぜ閉域網でグローバルユニキャストアドレスを割り振ったのか？

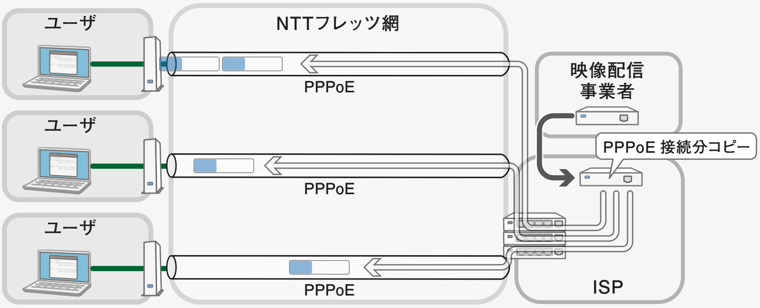
すでに触れたように、NTT NGNがIPv6インターネットとの接続性がない閉域網となっている背景には、NTT法による制限があります。では、なぜNTT東西では、その閉

†17 総務省「東日本電信電話株式会社及び西日本電信電話株式会社の第一種指定電気通信設備に関する接続約款の変更の認可 NGNのIPv6インターネット接続に係る接続約款の措置」

cc BY NC SA

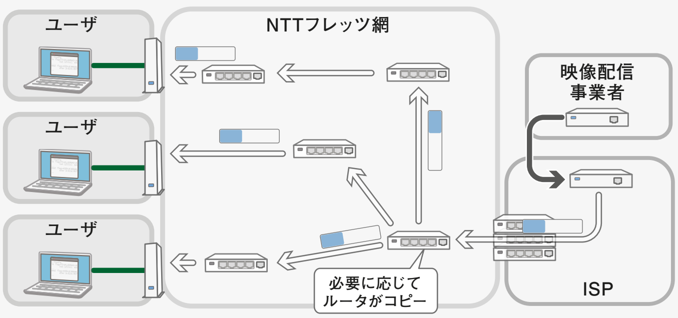
域網でユーザにグローバルユニキャストアドレスを割り振るというネットワークの構築を選択したのでしょうか？ その理由は、IPv6 マルチキャストを利用して映像配信をしたいというものでした[†18]。

NTT フレッツ網内で IPv4 マルチキャストによる映像配信を行った場合、図 A.18 のようにPPPoE を通じて各家庭まで IPv4 マルチキャストパケットが運ばれます。このとき、各映像データを運ぶ IPv4 マルチキャストパケットは、PPPoE によって集約された場所から各家庭宛に張られたトンネルへとコピーされたうえで送信されます。マルチキャストは、途中のルータがコピーを行ってくれるのでコストを削減できる仕組みです。PPPoE で集約している上流までしかコピーされず、それよりも下流では PPPoE セッションごとにすべてコピーしていたのでは、ユニキャストと変わらないトラフィック量になってしまって効率的な映像配信ができません。



▶ 図 A.18　IPv4 でマルチキャストを行う場合

IPv4 マルチキャストを利用したときに PPPoE を経由させる必要があるのは、当時のフレッツユーザが利用している IPv4 アドレスが ISP によって提供されているものであったためです。しかし、IPv6 マルチキャストを利用すれば、PPPoE を経由させずにユーザまで効率的に映像データを届けることが可能でした。IPv6 マルチキャストを利用した場合は、図 A.19 のような構成になります。



▶ 図 A.19　IPv6 でマルチキャストを行う場合

これは、IPv4 と IPv6 のプロトコルの違いというよりも、提供されている商用サービスの違いであったともいえます。当時、IPv6 サービスを提供している ISP はほとんど存在していませんでした。そのため、NTT 東西がユーザ宅にグローバルユニキャストアドレスを配ったとしても、IPv4 インターネットへの接続サービスを提供している ISP と直

接競合するとまでいえる状態ではありませんでした[†19]。そのような事情から、ISP が提供する IPv4 サービスと平行して、NTT 東西による IPv6 アドレスがユーザ宅に配布されました。

IPv4 と IPv6 がまったく別のプロトコルであり、それぞれまったく異なるネットワークトポロジであっても動作させることが可能であるという特性が、NTT 東西による映像配信の道を切り拓いたともいえます。IPv6 マルチキャストを利用したフレッツ網での映像配信サービスとして「ひかり TV」がありますが、2015 年に NTT 東西で 300 万を超えるサブスクライバー数であると発表されています[†20]。

なお、NTT 東西は地域会社として県間をまたぐ通信サービスを制限されていますが、2003 年以降、業務区域を越えない範囲、すなわち東は東、西は西の業務区域内では、当初は県内のみであった地域 IP 網の県間をつなぐ広域化が進められてきました。このような NTT 東西における広域化も、IPv6 マルチキャストサービスが企画される背景のひとつだったといえます。

[†18] http://www.geekpage.jp/blog/?id=2012/8/2/1

[†19] ただし、IPv6 から IPv4 へのフォールバック問題は発生するので、JAIPA での反対はありました。

[†20] http://www.nttplala.com/information/2014/3/20150311_2.html

［書籍版では空白のページです］

付録B

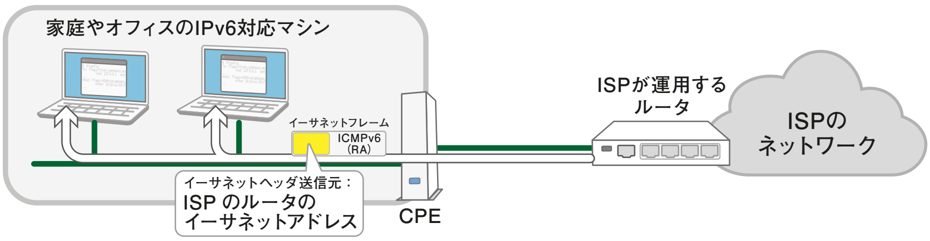
IPv6 パススルーと ND Proxy

家庭やオフィスで IPv6 インターネットへの接続サービスを利用する際、通常は ISP と契約して IPv6 プレフィックスを割り当ててもらうことになります。その際、海外では 8.7 節で説明した DHCPv6-PD が利用されるのが一般的です。しかし日本国内においては、「ISP のルータから Router Advertisement メッセージを直接提供する」ことにより IPv6 プレフィックスを割り当てる方法が広く採用されています。本付録では、そのための技術として利用されることが多い IPv6 パススルーと ND Proxy にて解説します。

B.1 ND パススルー

IPv6 では、ルータが Router Advertisement メッセージを別セグメントに転送することは禁止されています。そのため、ISP のルータからの Router Advertisement メッセージによってユーザに IPv6 プレフィックスを割り当てるには、ユーザの家庭内 LAN などのネットワークを「ISP のルータに直接つながったセグメント」として、ISP のルータからの Router Advertisement メッセージをユーザの機器が直接受け取れるようにする必要があります。それを実現する方法として、ユーザ側に設置される CPE で L2 スイッチと同じように IPv6 パケットを素通りさせるのが IPv6 パススルーです（図 B.1）。

IPv6 パススルーでは、家庭内にある機器が受け取るイーサネットフレームの送信元アドレスは、「ISP のルータのイーサネットアドレス」になります。



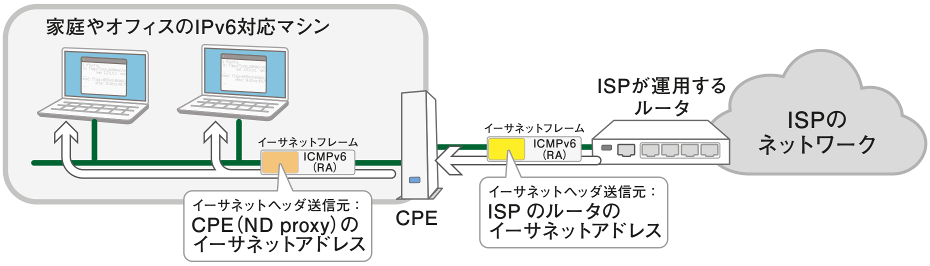
▶ 図 B.1　IPv6 パススルー

(cc) BY NC SA

B.2　ND Proxy

ND Proxyは、ExperimentalなRFCとしてRFC 4389で定義されています。ユーザ側に設置されるCPEがIPv6パケットを素通りさせるのではなく、ISPのルータから送信されるイーサネットフレームをND Proxyが一度受け取った上でプロキシとして転送します。L2で一度終端しつつ、L3の機能は持たずにプロキシとして動作するという方法です（図B.2）。ISPが運用するルータと家庭内とが、L2の観点では別セグメントになります。

ND Proxyでは、ユーザが受け取るイーサネットフレームの送信元アドレスが「家庭内にあるCPE（ND Proxyとして機能）のアドレス」になります。



▶ 図B.2　ND Proxy

RFC 4389には、ND Proxy機器がRouter Advertisementメッセージを転送するときに、そこに含まれるPビットを1にするという仕様があります。つまり、Pビットが1であるRouter Advertisementメッセージは、途中でND Proxyを経由したものであることがわかります。

Pビットが1になっているRouter AdvertisementメッセージをND Proxyで転送することは禁止されています。この仕様によってND ProxyによるRouter Advertisementメッセージのループを防ぐことができます。

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Type（134）	Code（0）						Checksum
Cur Hop Limit	M	O	H	Prf	P	予約	Router Lifetime
Reachable Time							
Retrans Timer							
Options ・・・							

▶ 図B.3　Router AdvertisementメッセージのPビット

NOTE

CPEによっては、IPv6パススルーとND Proxyのどちらかを選択できる機器もあります。それを切り替えつつ、流れているイーサネットフレームのヘッダを観察してみると、イーサネットの送信元アドレスが設定によって変化しているのを観測可能です。

B.3 IPv6パススルーやND Proxyとセキュリティ

IPv6パススルーは、文字どおりIPv6パケットを「そのまま通す」ので、機器の導入によりセキュリティが強化されることはありません。ND Proxyも、RFC 4389においてセキュリティを向上させるような特殊な機能が定義されているわけではありません。どちらも基本的にセキュリティを直接向上させる仕組みではないことに注意してください。

ただし、ND ProxyはL2のブリッジとして動作するので、何らかのフィルタとしての動作を実装しやすいという性質はあるかもしれません。実際、CPEによっては、ND proxyの設定を有効にすると自動的に外部からの通信に対して制限をつけるような機能が有効になる実装もあります。日本国内においてND Proxyが採用される背景にはこのような事情があるかもしれません。

［書籍版では空白のページです］

第1版へのあとがき

本書の企画を実現するにあたり、サポートしていただいた皆さまに感謝します。

GitHubでissueコメントをいただいた何人ものMakuakeサポーターの皆さまに感謝します。なかでも、技術的視点で数多くの詳細なフィードバックをいただいた佐藤広生さん、神明達哉さん、西田佳史さん、長谷部克幸さん、林達也さんに感謝します。DNSに関連する内容の間違いを指摘していただいた、浸透いうなさんに感謝します。フィードバック版を読んでいただけたものの特にコメントがなかった皆さまも、完成前のバージョンに目を通すために時間を割いていただき、ありがとうございました。

本書作成にあたり、応援のメッセージなどをいただいた皆さまに感謝します。

IETFでの積極的な議論や、kame実装など、IPv6の土台を作った故itojun（萩野純一郎）さんに感謝します。

Makuakeでの本書クラウドファンディング企画段階において、IPv6解説書の可能性を探るためにマクアケ社内のエンジニアばかりか社外でもITエンジニアと思しき人へ需要を調査するなど、非常に積極的かつ手厚いサポートをいただいたMakuakeの三田村さんに感謝します。

そして、最後に、電子版を無償配布したいという私の要望に応える形での出版をともに実現するためにご尽力いただいたラムダノート社の鹿野さんと高尾さんに感謝します。クラウドファンディングを行うための企画と運営、書籍の編集と出版を超高品質で行っていただきありがとうございます。特に、IPv6本のあるべき姿や、個別箇所に関する技術的な内容などを含めて議論を続け、一緒に本書を作り上げた鹿野さんは、もはや本書の共著者であるともいえるぐらいです。鹿野さんと一緒でなければ、本書を書き上げることはできなかったと考えています。

IPv6推進でも批判でもない、できるだけ中立的に技術や経緯に絞った情報発信を目指して本書を作りましたが、この本が少しでも多くの方々のお役に立てば幸いです。本書実現に協力いただいた皆さま、本当にありがとうございました。

2018年6月

小川晃通

CC BY NC SA

第2版へのあとがき

第1版に含まれていた間違いやわかりにくい点をご指摘いただいた皆さまに感謝します。なかでも、非常に多くのフィードバックをいただいた後藤芳和さんに感謝します。

第2版は、第1版と比べて「読みやすさ」が格段に上昇していると思います。第1版を出版してからYouTubeチャンネルでIPv6に関する情報発信を開始したのですが、そこで解説するために本書を参考にしながらコンテンツを作る過程で、本書の説明をもっと丁寧にできるのではないかと思い至りました。YouTube用の動画を作るたびに本書に説明を書き加える作業を繰り返しているうちに気づいたのは、本書が初心者にまったく優しくない構成だったことです。そこで第1部を大幅に改訂することにしました。第1部でIPv6の概要を説明することにより本書全体を見渡せるようにしたことが、第2版における最も大きな変化だと思います。

第2版では、第1版を出版した2018年以降に発生した各種RFCの更新にも対応しています。本を書き始める各段階で「IPv6関連仕様の変化や世の中の変化に対応するために本書を更新することを目指す」と宣言しました。おかげさまで、第2版として各種変化に対応する内容を追加し、更新を行うことができました。

IPv6に関連する仕様は、2021年現在も引き続き議論が続いており、今後もさまざまな変化が予想されます。本書でもっと詳しく解説したいと思いつつも実現できていない部分もまだあります。今後も本書の更新を目指したいと考えています。

引き続き、どうぞよろしくお願いいたします！

2021年12月

小川晃通

cc BY NC SA

クラウドファンディングにて本書第１版をご支援いただいた皆さま

（敬称略）

林 達也（@lef）/ Lepidum Co. Ltd.
藤本真樹（グリー株式会社）
石田慶樹
太一（https://github.com/taichi）
DeNA有志一同
渋川よしき、松木秀憲、
pospome、sairoutine、
小島泰洋、asakasa、
鴨志田良和、noppefoxwolf、
alarky、春山誠、立浪千尋、
中山貴博（@Civitaspo）、
田中貴洋、角田良太、
越智琢正、星野隼人、
sea_mountain、okitan、
戸上凌太、国賀由慎、
内海武夫、岡田健、吉田昌平、
持田章弘、高橋成人
ピクシブ株式会社
さくらインターネット株式会社
代表取締役社長 田中邦裕
株式会社朝日ネット
（Asahi Net, Inc.）
円佛公太郎
横山尚弥
Yukihiro Kikuchi
新村 信
木村成伴
三屋光史朗
兒島友三郎
（テックファーム株式会社）
saitara
佐原具幸
shirou
國武功一
浸透いうな
鈴木茂哉
宮川祥子
神明達哉
hkanemat
インフォコム（株）がねこまさし
Yugui
古庄道明
小林克志
草野貴之
加藤善規
瀧澤昭広
田村奈央
katsyoshi
@k1complete
入川幸之助
東平洋史
Perotan
秋斗
Toshiki Hayashi
UEM
サイボウズ株式会社
梅原 潤
Seiichi Kawamura
坂根昌一
土池政司
株式会社グラベルロード
HIROOKA Yasuo（hirooka.pro）
相原玲二
伊藤茂幸
mi-1aki
伊部祐久
峰野博史
山本 茂
三嶋将一郎
吉野純平
@muo_jp
くまもとゆたか
髙木正弘
阿部 博
塚田 学
佐々木 健
力武健次技術士事務所
村岡友介
だーはま
中村剛
川上雄也
もみじあめ（@momijiame）
長谷川明生
宇井太郎
十場 裕
mtera 寺西 功
滝澤隆史
樋口勝也
渡邊大記
b
小山哲志
深井宣之
objectx
佐藤 雄
八木澤健人
Rio Fujita
Taisuke Yamada
金子康行
日比野豊
蒲地輝尚
大脇 健
古畑幸之助
許 先明
コーダンス 小島慎太郎
川井浩陽
佐野健介（そな太）
湯谷啓明
西田佳史
宇夫陽次朗
森田邦裕
宮川大輔（@amedama）
坂本仁明
Toru Hisai
大竹和生
大本 貴
庖丁章浩
masahito
吉田真吾
藤田貴大
Buta_san
恵比澤賢
じゅんくどう
有賀征爾
オープンソース・ソリューション・テクノロジ（株） さとうふみやす
坂口俊文
sorah.jp
小川雄大
中村仁昭
藤田紀行
村上 学
長村善行
松崎吉伸
有限会社銀座堂
bongole
土屋太二
高橋成人
ちぇりお
まえだこうへい
Toshiki HAYASHI
村田がん
長谷川敏夫
kaorukit
sukiyaki project
さいとり
佐藤裕行
永山翔太
エリーツ合同会社
宇田周平
土本康生
富永崇之
shmz
清原智和
磯辺和彦
菊池希世識
@satokaz
鈴木健一
mewlist
平山陽一
きむらとものり
樋口敏広
遠藤陽平
松原弘典
中村貴志
宇多津真彦
@SRCHACK.ORG
安達昭人
村田耕司
中伏木新
小熊一輝
山本猛仁
鈴木祐耶
菅野健一
https://github.com/expf
::1
@rakusan
株式会社創夢 松山直道

森田悟史
Taketo Takashima
瀬見
鈴木裕亮
山田太郎
角田 裕
小河智章
渡部岳郎
丸橋弘明
Hideaki ODAGIRI
わーぷ
越智公王
vvakame
ゆたぽんさん
柳浦俊
渋谷直人
atrophy
befs_anne
横田真俊
stereocat
竹田幸生
bungoume
佐藤知津
よりいぬ
小倉 大
小林正幸
小嶋 智
重松光浩
佐藤秀樹
吉田大志（Hexa）
高木健介
横井雄太
大須賀義浩
zaki
北口修一
佐藤祐司
東村邦彦
id:Windymelt
海江田謙介
kitoku_magic
菊地謙
澤田石朋彦
濱野聖人
百々
水村直人
金城 雄
ヤリシンイチ
堀口祐司
西岡 学
安江 卓
南出成崇
寺岡慶佑
小林秀誉
岡本裕子
渡邊広志
新 善文
齋藤友誉
ティン@iMarket管理人
小池元洋
江本祐太
藤井一毅
Yasuo Ashina
Takumu Shiono
海老澤健太郎
副田哲生
besutome
お湯割り
Yoshinari Takaoka（a.k.a mumumu）
小山 勉
khrtz
野々垣裕司
中森 望
朝倉卓人
中島 滋
hakase109
小山海平
新美 bignum 誠
Rey.
kappapa
菊地俊介
ortus
菊池 久
ほりまさたけ
yosida95
ichii386
河野真知郎
安田 歩
瀬名波哲矢
Hiroyuki Ohno, Ph.D.（木いちごの会）
はやしさだたか
石部博之
水谷正慶
大平健司
末永洋樹
武樋彰浩
榑林勇気
Hirofumi Hida
三角 真
@ttsubo
伊藤孝一
中村計雄
後藤芳和（@goto_ipv6）
tockrock
大坪 力
伊藤琢巳
佐藤 治
松山裕樹
細野晃広
nyoro88
西 慶倫
さだまさつ
石田隆晃
うさみけんた（tadsan）
あふもん
加嶋啓章
山口孝穂
道向 新
Shigeichi Nakano
mafutaro
博士（びーる）
森 祥寛
そばゆ
鎌形システムエンジニアリング
内田泰広
岩佐 功
竹永吉伸
村山公保
木本雅彦@創夢
森下泰宏
若林寛樹
伊藤 崇
山田建史
前田貴之
fugahoge
並木秀人
藤澤昌隆
崎山洋輝
都築義仁
竹入寛章
shinya
吉田万浩
田中浩司
凛 火
sassembla
笠原 勉
今西幸一
山田英伸
大和田宗宏
のいず（@redwell_noise）
crark
toruuetani
阿部友範
前川 肇
lyokato
Tomokazu HARADA
eikihaya
神谷雅博
森本雅彦
櫻井達生
大原一真
イシクラヒロユキ
岩城理規
島崎清山（@seizans）
桑澤拓也
宮原
MarlTake
大津繁樹
西田晴彦
石山薫太郎
drillbits
池 徹
高山 温
山崎尚子
安江直樹
松山健吾
織 学
akiroom
横山優樹
佐藤弘明
株式会社レピダム
渡辺露文
畑谷進太
えいいち
uh
kgbu
伊達英之
tamuratak
えはらあい
米澤明生
小野田晃久

CC BY NC SA

吉田 史
上野栄実子
I.O
岡 啓
TAKAFUMI AMANO
耿 金洋
末廣雅利
山口慎也
北口将充
川辺治之
あらん
otsuka752
とうしみ
sige
ryuone
上田和志
長谷川喬也
tori_kizi
内田大樹
高石 潔
m-suzuki
中森康博
道正竜彦
今井志有人
津森洋伸
ainoniwa.net
鈴木真吾
Yusuke Sato
株式会社 Green Cherry
吉田真光
千田 霞
石井宏和
今井 潔
IMAZU Mitsumasa
伊藤敬彦
山本功司
Kei Hino
上松大輝（@hiroki_u）
ruicc
堀江正樹
higed
香月貴義
中村明彦
n_kane
佐藤広生
株式会社 YYOLO
山田幹彦
h-michael
土屋孝博
玉置一則
Tsuyoshi Tanaka
みるくてぃ
う゛ぁるちおん
岡 豊
馬篭修玄
Hideo Hattori
szktty
大川嘉一
園田真人
@nabeken
井野岡代之
寺岡良矩
たなむー
蛯名龍弘
tjmmm
高田雄一
yadokarielectric
島 慶一
堀米義仁
yutaro ikeda
そうたろう
高宮紀明
Yuuichi Akagawa
松原崇幸
中野和貴（かっきぃ）
東 真幸
たなかしんいち
大西尚利
@3socha
山田暁通
清水 順
okada
小寺康之
Makoto Kuwata
一色史雄
坂本龍亮
matche
Egoma
田中一直
加藤淳也
松浦佑紀
堤 清則
阪口琢哉（SAKAGUCHI, Takuya）
Liblib
江端ちはや
OMOIKANE
東海林健太
B21Soft, Inc.
板谷郷司
Yusei YAMAGUCHI
佐々木亮太
島 将人
layman
いわまぞん
岡田 洋
北海道大学 今井 適
株式会社 NT システムデザイン
自転車少年シンヤ君
綾田直輝
岡 勇佑
森 智哉
村田千尋
@seitaro1227
にしむらかずひろ
三嶋 晶
Hirotaka.Matsuo
板谷藍子
千田展也
高木省吾
二石真実
安部秀基
大野哲生
合同会社アペンド
門脇 優貴
たぼ <https://tabolog.net>
t.yamada
古本英男
鈴木亮一
埜中公博
岩澤昭一郎
小岩井航介
Yuzo Honda
水越雅浩
@philosophum
Kunihisa Takeda
眞城佳奈
竹内裕己哉
村井広一
水野稔晴
しろきゆうすけ
柚口佳予子
株式会社 NTT データ関西
山下真樹
Anxious
中村久美
児島孝典
菅原匡史
うなすけ
猿渡 敦
takabow
栗山 徹（@kotetu）
西 和人
@ke1ju
まてつ
大場聖史
naotaco
大越真弓
sinmetal
荒久田博士
河合 潤
野中啓弘
赤樹 将
馬淵俊弥
もりかずあき
篠宮俊輔
川原 啓
Tabuchi
鮫島康浩
ごまま
kabukawa
東瀬野丈史
たかはしまこと
齋藤将之
Toshi Aizawa
稲川雄一
加納憲男
村松 忍
中原祥宏
今井元太
北畠知行
河瀬大伸
後藤輝彦
Kinugasa
千葉周一郎
中野 岳
山根史嵩
情創技研（沖縄県中頭郡西原町）
NPoi
Naotsugu Okada
地藏真作
@s1061123
富安薫人

CC BY-NC-SA

倉科貴寛
中屋 誠
伊藤大悟
inu_on
酒井洋一
T.Hama
tagattie
Orion
池田直也
鴨川寛之
UC-XX
伊東英輝
Suza
関口裕士
徳差雄太
上坂エンリコ
みやこう
市橋一宣
Nefrock Inc.
Group Mary
Maffi Maffi
入谷和典
大鷲和紀
@toshis_
榎本博則
三上 洋
ながたゆうだい
しげじー
かわむらけいじ
助台良之
森福 茂
ST2
末光真理子

猪股秀樹
飯島昭博
とやまむねき
大原慎一郎
えづらあつし
新井良二
takumi kimura
Crosachi
tss
ダメ猿
@k-masatany
ぐり（@gurigoroblog）
村田祥弘
畑中貴弘
岡崎靖浩
柏崎央士
榮野隼一
Akira Charlie Hamano
結城さくら
南 敦史
永坂 智
岡 裕人
横地 晃
h.yoshioka
伊藤三喜男
Tadashi Morikubo
HONDA Yasuhiro
石川茂明
小野寺司史
望月裕之
魏 心宇
西田 剛
石川三晃

Kitamura Takehiko
たむらかずひこ
小澤 仁
ymdksk
よしはる
野田康夫
岡本義明
寺田雅史
大橋正和
白石 敦
JE3KMZ
syuu1228
さとうおさむ
たつみん
やっち
ぶのり
藤井勇太
酒見一幸
沖 勝
下竹章寛
植松秀文
澤田英祐
まみやなおき
toohide
梅木孝志
y-Aki
田村陽介
高橋祐也
藤原 豊
小松澤和弘
長谷部克幸

索引

記号・数字

*（QTYPE） 304, 320
.arpa トップレベルドメイン 321
.int トップレベルドメイン 321
.ip6.arpa. 310
/127（ルータ同士を接続） 72
/64 81
: 38
:: 39
::（RFC 5952） 41
::（未定義アドレス） 71
::/128 67
::/96 88
::[IPv4 アドレス] 88
::1/128 68
::ffff:[IPv4 アドレス] 86
[]（URI） 80
[]（ポート番号） 42
%（ゾーン識別子） 79
0100::/64 282
127.0.0.1 68
2000::/3 69
2001:db8::/32（例示用） 88
3ffe::/16（例示用としての） 155
464XLAT 430
4rd 404, 425, 429
　名前の由来 404
5f00::/8（例示用としての） 155
5 タプル 353
6LoWPAN Capability Indication オプション 146(表)
6LoWPAN Context オプション 146(表)
6man ワーキンググループ 33
6over4 402
6PE 405
6rd 402
　名前の由来 404
6rd Border Relay 403
6rd delegated prefix 404
6to4 386
　欠点 390
　デフォルト IPv6 アドレス選択 251
6to4 リレールータ 388

A

A+P 423
A6 322
AAAA
　データ表現 306
　登録のタイミング 319
AAAA フィルタリング 440
AAAA レコード 59, 303
AA フラグ 301
ACK 26
ADD 271
Additional セクション 304
Address Autoconfiguration 47, 128
Address plus Port 423
Address Registration オプション 146(表)
Address Resolution 47, 128
Address Selection オプション 255
Address-Specific Query 229
ADF 360
ADM 358
Advertise（DHCPv6 メッセージ） 187(表)
Advertisement 128
Advertisement Interval オプション 145(表)
AD フラグ 302
AF_INET6 284
AFRINIC 337
AFTR 421
AH 278
ALG 361, 408, 418, 433
All-Snoopers 235
All_DHCP_Relay_Agents_and_Servers 186
All_DHCP_Servers 186
ANCOUNT 302
Android 196
Answer セクション 304
ANY（QTYPE） 304
APDF 360
APDM 358
APNIC 337
Apple
　Happy Eyeballs 292
　iOS アプリ 412
ARCOUNT 302
ARIN 337
ARP 136
ARPA 310
ARPANET 4, 30
AS 20
ASM 232
Authentication Header 278
Authoritative Border Router オプション 146(表)
Authority セクション 304
AXFR 304
A レコード 59, 303

B

B4 421
BCP 38 243, 254
Best Current Practice 31
BGP 20
　マルチホーム 251
BGP4+ 20
Binding Request（STUN） 373
Bubble パケット 398

B フレッツ 448

C

CARD Reply オプション ..... 147(表)
CARD Request オプション ..... 147(表)
CATV 網 422
CDN
  6to4 391
  キャッシュ DNS サーバ 327
CD フラグ 302
Certificate オプション ..... 146(表)
Certification Path Advertisement メッセージ 272
Certification Path Solicitation メッセージ 272
Certification オプション 272
CE ルータ 85
CGA 268
CGA オプション ..... 146(表), 268
CGN 341, 362, 430
  アクセスログ 368
  課題 365
  機器に対する要件 364
CIDR 4
CLAT 431
Cone フラグ（Teredo） 393, 399
Confirm（DHCPv6 メッセージ） ..... 187(表)
CPE 349, 403, 446

D

DAD 161, 169
  エニーキャスト 241
  セキュリティ 265
  ループバック 172
DARPA 30
Decline（DHCPv6 メッセージ） ..... 187(表)
DELAY（近隣キャッシュのステート） 142
deprecated 160, 173–175, 246
Despres, Remi 405
Destination Address（Redirect） 144
Destination Address フィールド 95
Destination Unreachable メッセージ
  ..... 117(表), 119, 140, 216
  セキュリティ 121
DF ビット 216
DHCP Captive-Portal オプション ..... 146(表)
DHCP（IPv4 用） 181
  DHCPv6 との違い 180
  動作 181
DHCPv6 ..... 47, 159, 179
  Android 196
  動作例 183
DHCPv6-PD 48, 179, 201
DHCPv6 メッセージ 186
  Client Identifier オプション 190
  Server Identifier オプション 190
  オプション一覧 ..... 187(注)
  オプションのフォーマット 188
  オプション要求オプション 190
  基本フォーマット 186
  経過時間オプション 189
  ステータスコードオプション 188
Diffserv 94
`dig` 300
  AAAA レコードの例 308
  A レコードの例 308
  IPv4 逆引きの例 310
  IPv6 逆引きの例 311
DNS 59, 297
  Happy Eyeballs 288
  IPv4 と IPv6 を同時に問い合わせ 325
  IPv6 から IPv4 へのフォールバック 326
  サーバに関する情報の取得方法 180
  デュアルスタック 323
DNS Push Notification 290
DNS Recursive Name Servers オプション 195
DNS Search List オプション 195
DNS suffix search list 152
DNS-SD 378
DNS64 414
  読み方 ..... 411(注)
DNSSL オプション ..... 146(表), 152, 177
DNS サフィックス 152
DNS メッセージ 300
  512 オクテット問題 313
  セクション 300
  例 306
DPI 107, 280
DS Field 94
DS-Lite 421
DSCP 94
DUID 191
DUID-EN 192
DUID-LL 192
DUID-LLT 191
DUID-UUID 193
`DupAddrDetectTransmits` 169
Duplicate Address Detection 47, 128
Durand, Alain 422

E

Echo Reply メッセージ ..... 118(表), 124
Echo Request メッセージ ..... 118(表), 124
ECN 94
EDNS0 314
EIF 360
EIM 358
Encapsulating Security Payload ヘッダ ..... 107(表), 279
Enhanced DAD 171
ESP 279
EUI-48 形式 165
EXCLUDE フィルタ 232
Experimental 31
Extension & Upper-Layer ヘッダ 212

F

`fe80::/10` 68
`ff00::/8` 68
`ff02::1` 68
`ff02::1:2` 186
`ff02::2` 68
`ff05::1:3` 186
Final /8 Policy 344
Flags Expansion 133
Flags フィールド 101, 207
FLET'S 437
FLET'S.Net ..... 448(注)
Flow Label フィールド 94, 101
`Forwarded` 434
FQDN 300

CC BY NC SA

Fragment Offset フィールド 101, 207
FTP（IPv4/IPv6 変換） 418
Full Cone NAT 374, 392

G

gai.conf 250, 294
General Query 229
getaddrinfo() 246, 284, 294
gethostbyname() 286
GMA 426
Google over IPv6 330

H

Handover Assist Information オプション 146(表)
Handover Key Reply オプション 146(表)
Handover Key Request オプション 146(表)
Happy Eyeballs 288, 440
Header Checksum フィールド 101
Header セクション 301
hextet 42
Historic 31
Home Agent Information オプション 146(表)
homenet ワーキンググループ 251
Hop Limit フィールド 95, 99
Hop-by-Hop オプションヘッダ 107(表), 108
Host Identity Protocol 107(表)
HTTPS 343
HTTP プロキシ 433

I

IA_NA 198
IA_NA オプション 199
IA_PD 198
IA_PD オプション 202
IA_PD プレフィックスオプション 203
IA_TA 198
IA_TA オプション 200
IAB 89
IANA 35, 69, 337
IA アドレスオプション 199
ICANN 35
ICE 413
ICMP 115
  NAT 366
ICMPv6 46, 115
  API 284
  フィルタリング 279
ICMPv6 エラーメッセージ 117(表), 118
ICMPv6 情報メッセージ 118(表), 124
ICMPv6 メッセージ
  タイプ 117
  フィルタ 279
  フォーマット 116
Identification フィールド 101, 207
Identity Association 198
IETF 29
IGMP 115, 228
IGMPv3 228
IHL 212
IHL フィールド 99
IID → インターフェース識別子
in-addr.arpa 306(注)
INCLUDE フィルタ 232
INCOMPLETE 140
INCOMPLETE（近隣キャッシュのステート） 141
inet_ntop 286
inet_pton() 286
Information-request（DHCPv6 メッセージ） 187(表)
Informational 31
Internet Draft 30
invalid 173-175
iOS アプリ 412
IP Address/Prefix オプション 146(表)
IP6.ARPA 306
ip6.int 310
  廃止 321
IPoE 445
IPsec 263, 278
IPv4 NAT 347
IPv4-Compatible IPv6 アドレス 88
IPv4-Mapped IPv6 アドレス 41, 85, 248,250
  セキュリティ 87, 281, 331
  ソケット 292
IPv4/IPv6 共存技術 → 共存技術
IPv4/IPv6 変換機用 IPv6 アドレス 417
IPv4aaS 383
IPv4 アドレス 3
  IPv6 アドレスで表現 85
  アドレス空間 37
  アドレス長 37
  維持料 345
  移転 343
  構成要素 45
  在庫枯渇 4
  在庫枯渇への対策 340
  市場での売買 344
  返却 345
IPv4 互換 IPv6 アドレス 88
IPv4 射影 IPv6 アドレス . → IPv4-Mapped IPv6 アドレス
IPv4 の島 382
IPv4 ヘッダ 96
IPv6 3, 46
  IPv4 との違い 51
  イントラネット内部 83
  プライベートアドレス 83
IPv6 CE ルータ 85
IPv6 IPoE 257, 445
IPv6 Jumbogram 95, 317
IPv6 NAT 256
IPv6 PPPoE 257, 444
IPv6 PPPoE と IPv6 IPoE
  開始時期 451
  ビジネス構造 449
IPV6_V6ONLY 292
IPv6 アドレス
  IPv4/IPv6 変換機用 417
  IPv4 アドレスを含める 40
  URI で使う 80
  アドレス空間 37
  アドレス長 37
  暗号を使って生成 268
  大文字小文字 41
  実際に使える空間 38
  種類 67
  省略表現 39
  ステート 174(図), 176(図)
  設定例 48

選択 50
存続期間 173
テキスト表現 38
デフォルトアドレス 246
特殊用途 69
ノードに要求される〜 71
表記の名前 42
プライバシー 164, 275
マルチプレフィックス 154, 251
ユーザへの割り当て 89
利用モデル 44
IPv6 アドレス体系 67
概要 43
IPv6 移行技術 379
IPv6 インターネット 55
IPv6 拡張ヘッダ 105
API 284
順番 108
先頭 16 ビット 113
ファイアウォール 107
ルータで破棄 114
IPv6 サイトリナンバリング 257
課題 260
IPv6 島 62
IPv6 対応 54, 286
アプリケーションの 56
サーバの 57
ネットワークの 55
ユーザの 55
IPv6 ネットワーク 55
IPv6 フラグメンテーション
Atomic な 213
セキュリティ 275
例 210
IPv6 ヘッダ 5, 93
IPv6 への移行 54, 62
IP アドレス 3
文字列に変換 286
IP エニーキャスト
TURN 378
IP フラグメンテーション 205
IS-IS 20
ISATAP 399
脆弱性 401
ISATAP IPv6 アドレス 400
ISO 29
ISP Shared アドレス 368
ITU 29

J

Joke RFC 31
JPNIC 338

L

L2 スイッチ 23
L3 スイッチ 23
L7 スイッチ 23
LACNIC 337
Large Scale NAT 362
lifetime（NAT） 355
Lightweight 4over6 422
Lightweight-MLDv2 233
Link-layer Address オプション 146(表)
LIR 338
LSB 7
LSN 362
LTE 27
LW-MLDv2 233
lw4o6 422
lwAFTR 422
lwB4 422

M

MAC アドレス
インターフェース識別子 165
例示用 88
MAN 362
MAP BR 426
MAP CE 426
MAP-E 425
MAP-T 425
MAP オプション 146(表)
mDNS 378
MLD 228
MLDv2 118(表), 228
MLD スヌーピング 235
MLD メッセージ 229, 230
DAD 171
Mobile Node Identifier オプション 146(表)
Modified EUI-64 形式 164
MRD 235
MSB 7
MSS 207, 218
MTU 205, 215
MTU オプション 145(表), 150
Multicast Listener Done メッセージ 118(表), 230
Multicast Listener Query メッセージ 118(表),229
Multicast Listener Report メッセージ 118(表),230
Multicast Router Advertisement メッセージ 118(表),235, 236
Multicast Router Solicitation メッセージ 118(表),235, 236
Multicast Router Termination 235
Multicast Router Termination メッセージ 118(表),236
Multiple Address NAT 362

N

NA 129,→ Neighbor Advertisement メッセージ
NAPT 348
NAT 4, 341, 347
IPv6 256
IPv6 における必要性 90
RFC 3489 での分類 392
機器に要求される挙動 356
旧 STUN における分類 373
同時通信セッション数の上限 363
ユースケース 350
NAT binding 352
NAT mapping 352(注)
NAT-PT 408
NAT44 408
NAT444 362
NAT64 408, 411, 413
ステートフル 412
読み方 411(注)
NAT64+DNS64 411
NAT66 90, 445
NAT 越え 366

CC BY NC SA

NAT ルータ 349
NBMA Shortcut Limit オプション ......145(表)
ND ......→ 近隣探索プロトコル
NDP 127
Neighbor Advertisement Acknowledgment オプション ......146(表)
Neighbor Advertisement メッセージ ......118(表), 137
DAD 170
近隣不到達性検知 140
脆弱性 156
フォーマット 138
リンク層アドレス解決 139
Neighbor Solicitation メッセージ ......118(表), 136
DAD 170
近隣不到達性検知 140
フォーマット 136
リンク層アドレス解決 139
Neighbor Unreachability Detection 47, 128
New Router Prefix Information オプション ......146(表)
Next Header フィールド 94, 100, 106
Next-hop Determination 47, 128
NGN 448
NIR 338
No Next Header 112
Nonce オプション ......146(表), 271
NPT 251
NPTv6 ......83, 90
NS ......129, → Neighbor Solicitation メッセージ
NSCOUNT 302
nslookup 300
NS レコード 303
NTT 451
NTT NGN 257, 437
NTT NGN IPv6 マルチプレフィックス問題 437
NTT フレッツ網 437
NTT 閉域網 IPv6 フォールバック問題 437
NTT 法 451
NUD 140
NwGN ......448(注)

O

off-link 154
on-link 148, 154
on-link assumption 157
on-link エニーキャスト 240
optimistic 175
Optimistic DAD 175
Option Type フィールド 110
OPTION_CLIENTID 190
OPTION_ELAPSED_TIME 189
OPTION_ORO 190
OPTION_SERVERID 191
OPTION_STATUS_CODE 188
OSI 参照モデル 23
OSPF 19
OSPFv3 19
OUI 165

P

Packet Too Big メッセージ ......117(表), 121, 217
最小 MTU 214
Pad1 111
PadN 111
Parameter Discovery 47, 128
Parameter Problem メッセージ ......117(表), 123
Path MTU Discovery 215
問題 218
PA アドレス 345
PCP 365
Per-Fragment ヘッダ 211
PF_INET6 284
PIM 234
PIM-DM 234
PIM-SM 234
PI アドレス 345
PLAT 431
POI 449
Policy Table オプション 255
Port Restricted Cone 361
Port-restricted Cone NAT 375, 393
PPPoE 444
preferred 160, 173–175
Preferred Lifetime 149
preferred-lifetime 174, 199
Prefix Discovery 47, 128
Prefix Exclude オプション 204
Prefix Information オプション ......145(表), 148, 155
PROBE（近隣キャッシュのステート） 142
Protocol フィールド 100
Proxy Signature オプション ......146(表)
PRR 423
PSID 426
PTR 305, 310

Q

QDCOUNT 302
QNAME フィールド 303
QTYPE フィールド（Question セクション） 303
Question セクション 303
QUIC 27

R

RA ......128, → Router Advertisement メッセージ
RA ガード 273
RA フラグ 302
RCODE 302
RDNSS オプション ......146(表), 151, 177
RD フラグ 298, 302
REACHABLE（近隣キャッシュのステート） 142
ReachableTime（近隣不到達性検知） 143
Rebind（DHCPv6 メッセージ） ......187(表)
Reconfigure（DHCPv6 メッセージ） ......187(表)
Redirect 47, 128
Redirected Header オプション ......145(表), 150
Redirect メッセージ ......118(表), 143, 156
フォーマット 143
Relay-Forward（DHCPv6 メッセージ） ......187(表)
Relay-Reply（DHCPv6 メッセージ） ......187(表)
Release（DHCPv6 メッセージ） ......187(表)
Renew（DHCPv6 メッセージ） ......187(表)
Reply（DHCPv6 メッセージ） ......187(表)
Request（DHCPv6 メッセージ） ......187(表)
Restricted Cone NAT 374, 393
RFC 29
IPv6 の基本仕様 ......34(図)
各章の初出
RFC 1034 ......297, 325

RFC 1035 153, 195, 297, 313,325
RFC 1071 103
RFC 1112 221
RFC 1122 45
RFC 1123 317
RFC 1149 31
RFC 1191 208, 215
RFC 1347 98
RFC 1475 98
RFC 1597 349
RFC 1621 98
RFC 1631 348
RFC 1671 287
RFC 1883 iii, 33
RFC 1884 43, 240
RFC 1886 306
RFC 1918 250
RFC 1981 215
RFC 1996 301
RFC 2026 33
RFC 2072 258
RFC 2080 19
RFC 2081 19
RFC 2113 232
RFC 2131 179
RFC 2136 301
RFC 2181 297
RFC 2365 225
RFC 2373 43, 240
RFC 2428 418
RFC 2460 32, 102, 212
RFC 2462 173
RFC 2464 24, 96, 148, 237
RFC 2473 385
RFC 2474 94
RFC 2491 145
RFC 2526 242
RFC 2529 402
RFC 2543 361
RFC 2606 89
RFC 2663 348
RFC 2671 314
RFC 2673 314
RFC 2675 95
RFC 2710 118, 229
RFC 2765 86, 409
RFC 2766 408
RFC 2782 328
RFC 2826 60, 323
RFC 2827 243, 254, 280
RFC 2893 392
RFC 2894 258
RFC 3022 348
RFC 3027 409
RFC 3056 386
RFC 3068 390
RFC 3122 146
RFC 3142 434
RFC 3168 94
RFC 3177 89
RFC 3226 314
RFC 3258 239
RFC 3260 94
RFC 3306 223
RFC 3307 224
RFC 3319 180, 196
RFC 3397 154
RFC 3424 359
RFC 3425 301
RFC 3484 250
RFC 3489 373, 392
RFC 3493 86, 284
RFC 3513 43, 240
RFC 3542 284
RFC 3590 233
RFC 3596 305
RFC 3627 74
RFC 3646 152, 195
RFC 3678 284
RFC 3692 107, 147
RFC 3697 102
RFC 3704 243
RFC 3756 155, 265
RFC 3775 132
RFC 3810 118, 232
RFC 3849 88
RFC 3879 83, 250
RFC 3882 281
RFC 3898 180
RFC 3901 320, 324
RFC 3904 392
RFC 3927 76, 250
RFC 3956 224
RFC 3964 391
RFC 3971 146, 267
RFC 3972 268
RFC 3973 234
RFC 3986 42
RFC 4007 71, 226
RFC 4035 302
RFC 4038 56, 86, 284, 331
RFC 4065 147
RFC 4068 146
RFC 4086 82
RFC 4122 193
RFC 4140 146
RFC 4191 132
RFC 4192 173, 258
RFC 4193 82, 251
RFC 4213 382, 385
RFC 4286 118, 235
RFC 4291 39, 67, 157, 163,224, 240, 418
RFC 4294 96, 279
RFC 4301 278
RFC 4302 107, 279
RFC 4303 107, 279
RFC 4339 180
RFC 4380 392
RFC 4389 132
RFC 4429 173
RFC 4443 46, 116, 130
RFC 4472 319, 324
RFC 4541 236
RFC 4604 233
RFC 4607 232
RFC 4632 4
RFC 4659 86
RFC 4727 107, 147

RFC 4760 20
RFC 4782 107
RFC 4786 239
RFC 4787 .... 356, 374, 413
RFC 4798 .... 86, 405
RFC 4821 218
RFC 4844 32
RFC 4845 32
RFC 4846 32
RFC 4861 .... 118, 127, 160, 226,265
RFC 4862 .... 75, 127, 160, 233,241
RFC 4864 .... 90, 362
RFC 4890 .... 222, 279
RFC 4903 155
RFC 4941 167
RFC 4942 87
RFC 4966 409
RFC 5006 .... 153, 177
RFC 5014 284
RFC 5095 107
RFC 5128 366
RFC 5135 367
RFC 5158 392
RFC 5175 133
RFC 5214 400
RFC 5237 94
RFC 5245 413
RFC 5269 146
RFC 5271 146
RFC 5308 20
RFC 5340 19
RFC 5375 242
RFC 5382 .... 356, 413
RFC 5389 373
RFC 5398 89
RFC 5453 242
RFC 5508 .... 364, 413
RFC 5533 107
RFC 5568 146
RFC 5569 402
RFC 5626 373
RFC 5635 281
RFC 5722 213
RFC 5737 88
RFC 5742 32
RFC 5766 377
RFC 5790 233
RFC 5838 19
RFC 5855 310
RFC 5871 112
RFC 5887 258
RFC 5902 84
RFC 5908 180
RFC 5927 218
RFC 5942 156
RFC 5952 39
RFC 5966 317
RFC 5969 403
RFC 5991 399
RFC 6052 408
RFC 6081 399
RFC 6085 237
RFC 6104 272
RFC 6105 272
RFC 6106 .... 153, 160, 196
RFC 6144 407
RFC 6145 408
RFC 6146 408
RFC 6147 408
RFC 6156 377
RFC 6157 435
RFC 6164 74
RFC 6169 280
RFC 6177 89
RFC 6180 382
RFC 6195 297
RFC 6250 91
RFC 6264 430
RFC 6269 366
RFC 6275 .... 107, 145
RFC 6294 103
RFC 6296 83
RFC 6302 368
RFC 6308 223
RFC 6333 421
RFC 6343 390
RFC 6346 424
RFC 6355 191
RFC 6384 409
RFC 6398 232
RFC 6410 33
RFC 6434 .... 96, 279
RFC 6436 102
RFC 6437 102
RFC 6496 146
RFC 6540 286
RFC 6547 74
RFC 6554 112
RFC 6555 290
RFC 6556 292
RFC 6564 113
RFC 6586 416
RFC 6589 .... 57, 329
RFC 6598 368
RFC 6603 204
RFC 6666 282
RFC 6724 .... 176, 246, 289, 386
RFC 6732 392
RFC 6761 89
RFC 6762 378
RFC 6763 378
RFC 6775 146
RFC 6864 207
RFC 6866 258
RFC 6874 80
RFC 6877 431
RFC 6879 258
RFC 6883 57
RFC 6887 365
RFC 6888 363
RFC 6890 69
RFC 6891 314
RFC 6935 105
RFC 6936 105
RFC 6946 213
RFC 6970 366
RFC 6980 267
RFC 7010 258

CC BY NC SA

RFC 7021 ... 367
RFC 7042 ... 88
RFC 7045 ... 107
RFC 7078 ... 252
RFC 7084 ... 85
RFC 7094 ... 240
RFC 7112 ... 211, 275
RFC 7113 ... 272
RFC 7136 ... 166
RFC 7157 ... 90, 252, 441
RFC 7217 ... 164, 277
RFC 7227 ... 188
RFC 7239 ... 434
RFC 7346 ... 225
RFC 7368 ... 84
RFC 7381 ... 263
RFC 7400 ... 146
RFC 7401 ... 107
RFC 7421 ... 81
RFC 7526 ... 390
RFC 7527 ... 171, 271
RFC 755 ... 98
RFC 7596 ... 422
RFC 7597 ... 423
RFC 7598 ... 423
RFC 7599 ... 425
RFC 7600 ... 425
RFC 7610 ... 275
RFC 768 ... 103
RFC 7707 ... 169, 267
RFC 7710 ... 146
RFC 7721 ... 169, 276
RFC 7761 ... 234
RFC 7766 ... 319
RFC 7857 ... 356
RFC 7872 ... 114, 214
RFC 791 ... 9, 54, 205
RFC 7915 ... 408
RFC 793 ... 103
RFC 8021 ... 213
RFC 8028 ... 256
RFC 8064 ... 164, 226
RFC 8106 ... 133, 160, 196
RFC 8155 ... 378
RFC 8168 ... 203
RFC 8200 ... 12, 46, 93, 117, 211, 275
RFC 8201 ... 215
RFC 8215 ... 418
RFC 8305 ... 288
RFC 8415 ... 179
RFC 8445 ... 373
RFC 8482 ... 320
RFC 8489 ... 372
RFC 8501 ... 312
RFC 8504 ... 96, 160, 279
RFC 8585 ... 383
RFC 879 ... 208
RFC 8981 ... 92, 164, 200, 275
RFC 9000 ... 27
RFC 959 ... 361, 418
名前の由来 ... 30
RIP ... 19
RIPE NCC ... 337
RIPng ... 19
RIR ... 69, 337
Root ... 60
Route Information オプション ... 146(表)
Router Advertisement Flags Extension オプション ... 146(表)
Router Advertisement メッセージ ... 118(表), 130, 172
DNS 情報 ... 177
セキュリティ ... 154, 272
フォーマット ... 131
フラグ ... 133
プレフィックス情報発見 ... 129
ルータ発見 ... 129
Router Alert オプション ... 231
Router Discovery ... 47, 128
Router Solicitation メッセージ ... 118(表), 133, 173
フォーマット ... 133
Routing Type ... 112
RPF ... 243
RR ... 304
RS ... 129, → Router Solicitation メッセージ
RSA Signature オプション ... 146(表), 269

S

S-NAPTR ... 378
SAM ... 425
SCTP ... 27
SEND ... 267
SFI ... 362
Shared Secret（STUN） ... 373
Shim6 Protocol ... 107(表)
SIIT ... 410, 425
NAT64 ... 413
制約 ... 411
SIP ... 361, 435
DHCPv6 ... 196
SLAAC ... 47, 159, 160
名前の由来 ... 173
利用例 ... 161
SLAC ... 173
SNI ... 343
`sockaddr_in6` ... 284
`sockaddr_storage` ... 284
Socket API ... 284
アプリケーションの動作 ... 285
Softwire46 ... 423
softwire ワーキンググループ ... 425
SOHO ルータ ... 349
Solicit（DHCPv6 メッセージ） ... 187(表)
Solicitation ... 129
Solicited-Node マルチキャストアドレス ... 225
Solicited-Node マルチキャストグループ ... 139, 171
Source Address List オプション ... 146(表)
Source Address フィールド ... 95
Source Link-layer Address オプション ... 145(表), 147
SPF ... 362
SRV ... 305, 328
SSM ... 232
Stable address ... 49, 168
STALE（近隣キャッシュのステート） ... 142
Standards Track ... 31
STUN ... 371
新旧 ... 372
Symmetric NAT ... 376, 393

CC BY NC SA

T

Target Address List オプション 146(表)
Target Address（Redirect） 144
Target Link-layer Address オプション 145(表), 148
TCP 22, 25
　Path MTU Discovery 218
　エニーキャスト 240
　チェックサム計算 103
TCP RST 438
TCP/IP 26
TCP フォールバック（DNS メッセージ） 317
TC フラグ 301, 317
Temporary address 168(注)
Temporary IPv6 アドレス 49, 92, 164, 166, 275
　DNS 登録 319
　デフォルト IPv6 アドレス選択 250
tentative 161, 173, 175
Teredo 392
　問題点 395
Teredo IPv6 アドレス 398
Teredo リレー 393
Time Exceeded メッセージ 117(表), 122
Timestamp オプション 146(表), 270
TLV 形式 110
ToS フィールド 99
`traceroute` 122
Traffic Class フィールド 94
TRT 434
Trust Anchor オプション 146(表), 272
TTL フィールド 99
TURN 377
Type 0（ルーティングヘッダ） 112
TYPE フィールド（リソースレコード） 305

U

UDP 22
　ゼロチェックサム 104
　チェックサム計算 103
ULA 82
　グローバル ID の生成アルゴリズム 83
　スコープ 71
　デフォルト IPv6 アドレス選択 251
universal/local ビット 165
UNSAF 359
UPnP IGD 365
URI で IPv6 アドレスを使う 80
uRPF 243(注)
u ビット 165

V

Valid Lifetime 149
valid-lifetime 174, 199
VNE 443, 445

W

World IPv6 Launch 330

X

`X-Forwarded-For` 434
XLAT 410

ア

アック 26
宛先オプションヘッダ 107(表), 109
アドミンローカルスコープ 224
アドレス 3
アドレス依存フィルタリング 360
アドレス依存マッピング 358
アドレス空間 37
アドレス自動設定 159
アドレススキャニング 276
　IPv6 サブネット 278
アドレス長 37
アドレスとポート依存フィルタリング 360
アドレスとポート依存マッピング 358
案 2 451
案 4 451
暗号
　IPv6 アドレス 268
　ペイロードの 279

イ

イーサネット
　IPv6 パケットの扱い 24, 96
　フレーム 23
　マルチキャスト 237
移行技術 61
位置トラッキング 276
インターネットプロトコル 10
インターフェース識別子 13, 45
　自動生成 164
インターフェースローカルスコープ 224

エ

エイプリルフール RFC 31
エニーキャスト 43, 239
　6to4 389
　DAD 162, 170
エンディアン 7
エンドノード 12
エンドポイント非依存フィルタリング 360
エンドポイント非依存マッピング 358

オ

オクテット 5(注)
折り返し通信 364, 446

カ

回線交換方式 4
角括弧 42, 80
拡張ヘッダ → IPv6 拡張ヘッダ
カプセル化 385
　6to4 387
仮（IPv6 アドレス） 173

キ

疑似 IPv6 ヘッダ 104, 117
疑似ヘッダ 103
疑似レコード（EDNS0） 316
逆引き（DNS） 309
　IPv6 における問題 311
キャッシュ DNS サーバ 298
共存技術 61, 379
　運用 421
　分類 381
近隣キャッシュ 139

CC BY NC SA

ステート 141
近隣探索
Neighbor Advertisement メッセージ 137
Neighbor Solicitation メッセージ 136
近隣探索プロトコル 46, 127
近隣不到達性検知 136
セキュリティ 265
プレフィックス情報発見 129, 134, 135
リンク層アドレス解決 136
ルータ発見 129, 134, 135
近隣探索メッセージ 128
オプション 145
オプションのフォーマット 147
近隣不到達性検知 47, 128, 136,140
Neighbor Advertisement メッセージ 140
Neighbor Solicitation メッセージ 140

ク

クエリ（DNS） 301
組み合わせによる運用手法 382
クラス（リソースレコード） 304
グルー 299
グローバルアドレス 161
自動生成 172
マルチプレフィックス 254
グローバルスコープ 71, 225
グローバルユニキャストアドレス 68, 71,80
NTT NGN 438
割り振り 69
グローバルルーティングプレフィックス 80

ケ

経路情報 12
権威サーバ 299
検索ドメイン 152

コ

コイントス 429
混在環境 54

サ

再帰問い合わせ 298
サイトリナンバリング → IPv6 サイトリナンバリング
サイトローカルアドレス 71, 83
デフォルト IPv6 アドレス選択 250
廃止の理由 83
サイトローカルスコープ 224
サブネット 12
モデル（RFC 5942） 156
サブネットプレフィックス 13, 45
サブネットルータエニーキャストアドレス 72, 241

シ

識別情報（通信の） 102
次世代インターネットプロトコル 33
思想
IPv4/IPv6 共存技術 381
IPv6 ヘッダに見る 96
IP サービスの前提 91
インターネットにおける分散処理 27
次ホップ 10
十六進表記 38
情報メッセージ（ICMPv6） 118(表)
商用インターネット接続サービス 348
自律システム 20

ス

推奨（IPv6 アドレス） 160
スコープ 71
スコープ（マルチキャストの） 224
ステート（IPv6 アドレスの） 174(図),176(図)
ステートフル DHCPv6 179, 196
動作例 184
ステートフル NAT64 412
ステートフルなプロトコル 382
ステートレス DHCPv6 179, 193
動作例 183
ステートレスアドレス自動設定 159
ステートレスなプロトコル 382
ストリーム 32

セ

生存期間（セッション） 366
生存時間（NAT） 355
正引き（DNS） 309
セキュリティ
6to4 391
DAD 265
IPv6 263
近隣探索プロトコル 265
セクション 300
セグメント 207
ゼロチェックサム 104
全ノードアドレス 68
全ルータアドレス 68

ソ

送信元 IPv6 アドレス
Happy Eyeballs Version 2 289
エニーキャストアドレス 240
デフォルトの選択ルール 247
マルチホーム 255
送信元 IP アドレス
HTTP セッションの 434
ゾーン 71, 76
マルチキャスト 226
ゾーン ID 79
ゾーンインデックス 79
ゾーン識別子 79
ソケット 261
IPv6 と IPv4 を両方扱う 86
組織ローカルスコープ 224
存続期間 160

チ

地域 IP 網 451
チェイニング 322
チェックサム 103
ICMPv6 117
NAT での再計算 355
UDP 104
ゼロ〜 104
重複アドレス検知 161

CC BY NC SA

テ

- テキスト表記
  - ゾーンインデックス 79
- テキスト表記（IP アドレス） 38
- デフォルト IPv6 アドレス選択 246
  - API 284
  - Happy Eyeballs 289
- デフォルトゲートウェイ 18（注）
- デフォルトのドメイン名 152
- デフォルトルータ 18
- デュアルスタック 58, 323
  - Happy Eyeballs 288
  - アプリケーション 331

ト

- 匿名アドレス → Temporary IPv6 アドレス
- ドット付き十進表記 38, 42
- トポロジ 17
- トランスポート（DNS メッセージ） 323
- トランスポート層 22, 25
- トンネル技術 381
  - セキュリティ 280, 405

ナ

- 名前 59
- 名前解決 59
- 名前空間 323
  - フラグメント化 321

ニ

- 認証委任検索 271
- 認証ヘッダ 107（表）, 279

ネ

- ネットマスク 13
- ネットワーク 12
- ネットワークアプリケーション 283
- ネットワーク層 21
- ネットワークバイトオーダー 9
  - RFC 9
- ネットワーク部 13, 45

ノ

- ノード 12
  - 要求される IPv6 アドレス 71

ハ

- バーチャルサーキット 26
- バーチャルホスト 342
- バイト 5
- バイトオーダー 7
- パケット 4
  - 入れ子としての 23
  - バイト列としての 5
- パケット交換方式 4
- パディング 111
- 反復問い合わせ 299

ヒ

- ひかり TV 453
- 非再帰問い合わせ 299
- 非推奨（IPv6 アドレス） 160, 246
- ビッグエンディアン 7
- ビットマスク 13
- ピンポン状態（パケットの） 72

フ

- ファイアウォール 280
- フィールド 6
- フィルタリング 357
- フォーマット 5
  - バイトオーダー 9
- フォールバック
  - IPv6 から IPv4 への 287, 326, 438
- フォワーディング 16
- プライバシー 91, 164
  - IPv6 アドレス 275
- プライバシーアドレス
  - 319, → Temporary IPv6 アドレス
- プライベート IPv4 アドレス 4, 349, 369
  - ISP 用 368
- プライベートアドレス
  - IPv6 83
- フラグメンテーション → IPv6 フラグメンテーション
- フラグメンテーション（IPv4） 216
- フラグメント化不能部分 212
- フラグメント識別子 210
- フラグメントヘッダ 107（表）, 208
  - RFC 6564 113
- ブラックホール 287
- ブラックホール用 IPv6 アドレス 281
- ブラックリスティング（IPv6 DNS） 329
- フレッツ 437
- フレッツ光ネクスト 448
- プレフィックス 15
- プレフィックス情報発見（近隣探索） 129
  - ホストの動作 135
  - ルータの動作 134
- プレフィックス除外オプション 204
- プレフィックス長 15, 44, 81
- フロー 102, 351
- ブロードキャスト 221
- ブロードバンドルータ 349
- フローラベル 102, 217
- プロキシ方式 433
- プログラミング 283
- プロトコル 3
- プロトコルスタック 58
- プロトコルパラメーター一覧 36
- プロファイリング 281

ヘ

- ヘアピニング 364
- ペイロード 5
- ヘッダ 5
- ヘッダフォーマット 5
- 変換技術 382, 407

ホ

- ポート番号
  - IPv4 アドレス在庫枯渇対策 342
  - 表記 41
- ホスト 12

エニーキャスト......240
ホストバイトオーダー......9
ホスト部......13, 45
ポリシーテーブル......249
実装......294
ホワイトリスティング（IPv6 DNS）......329

マ

マスク......13
マッピング......357
マルチアドレス......245
マルチキャスト......43
ソケットオプション......284
閉域網......452
リンク層......236
ルータ越え......234
マルチキャストアドレス......68, 221
構造......222
マルチキャストグループ......221
マルチキャスト配信ツリー......234
マルチキャストルータ......234
マルチプレフィックス......245, 251
NTT NGN......440
次ホップ......256
問題......253
マルチホーム......50, 251
シナリオ1......252
シナリオ2......252
シナリオ3......253
マルチリンクサブネット......157

ミ

未定義アドレス......67
スコープ......71

ム

無線LANルータ......349

モ

モビリティヘッダ......107(表)

ユ

ユニークローカルIPv6ユニキャストアドレス.....→ ULA
ユニキャスト......43
アドレスの構成要素......45

ラ

ラベル（QNAME）......303
ランデブーポイント......224, 234

リ

リソースレコード......304
リトルエンディアン......7
リンク層......21
リンク層アドレス解決......147
Neighbor Advertisement メッセージ......139
Neighbor Solicitation メッセージ......139
近隣探索......136
リンクローカルアドレス......75
IPv4......76
アドレス自動生成......162
リンクローカルスコープ......71, 224
リンクローカルユニキャストアドレス......44, 68,75
プレフィックス長......75

ル

ルータ......10, 12, 72
IPv6拡張ヘッダを破棄......114
アドレス自動設定......161
要求されるIPv6アドレス......72
ルータID......20
ルータ発見（近隣探索）......129
セキュリティ......271
ホストの動作......135
ルータの動作......134
ルーティングテーブル......10
ルーティングプロトコル......17, 19
ルーティングヘッダ......107(表), 111
Type 0......112
ルートDNSサーバ......299
IPv6トランスポート......324
ルートサーバ......35
IPv4エニーキャスト......239
ループバック（DAD）......172
ループバックアドレス......68
ループバックインターフェース......71

レ

例示（インターフェース識別子）......165
例示用アドレス......88, 155
レイヤー2......23
レイヤー3......23
レルムローカルスコープ......224

ワ

割り当て......339
割り振り......339

■ 著者紹介

小川晃通（おがわ あきみち）

慶應義塾大学にて博士（政策・メディア）取得。各種TCP/IP解説書執筆。書籍執筆やライターとしての仕事の他に、IT系エンジニアとしてプログラミングでの受託開発や、コンサルティング、講演等も行う。慶應義塾大学にて博士（政策・メディア）取得。『インターネットのカタチ』『マスタリングTCP/IP OpenFlow編』（オーム社）、『アカマイ 知られざるインターネットの巨人』（KADOKAWAメディアファクトリー）、『ポートとソケットがわかればインターネットがわかる』（技術評論社）、『Linuxネットワークプログラミング』（SBクリエイティブ）、『徹底解説v6プラス』『ピアリング戦記 — 日本のインターネットを繋ぐ技術者たち』（ラムダノート）の著者/共著者。

YouTubeチャンネル https://www.youtube.com/user/geekpage

技術系ブログ「Geekなページ（http://www.geekpage.jp/）」管理人。

cc BY NC SA

技術書出版社の立ち上げに際して

コンピュータとネットワーク技術の普及は情報の流通を変え、出版社の役割にも再定義が求められています。誰もが技術情報を執筆して公開できる時代、自らが技術の当事者として技術書出版を問い直したいとの思いから、株式会社時雨堂をはじめとする数多くの技術者の方々の支援をうけてラムダノート株式会社を立ち上げました。当社の一冊一冊が、技術者の糧となれば幸いです。

鹿野桂一郎

プロフェッショナルIPv6 第2版

Printed in Japan ／ ISBN 978-4-908686-11-5

2018年 7 月20日　第1版第1刷 発行
2021年12月20日　第2版第1刷 発行
2023年 3 月 1 日　第2版第2刷 発行

著　者 小川晃通
発行者 鹿野桂一郎
編　集 高尾智絵
制　作 鹿野桂一郎
装　丁 凪小路
印　刷 三美印刷
製　本 三美印刷

発　行 ラムダノート株式会社
lambdanote.com
所在地 東京都荒川区西日暮里 2-22-1
連絡先 info@lambdanote.com

CC BY NC SA